AD-A091 653 ARMY CONCEPTS ANALYSIS AGENCY BETHESDA MD F/G 9/2
PROGTEST: A COMPUTER SYSTEM FOR THE ANALYSIS OF COMPUTATIONAL C--ETC(U)
APR 80 S BRAVY
UNCLASSIFIED CAA-D-80-1 NL

1 OF 2
AD A
091653

DISCLAIMER

The findings in this document are not be construed as an official Department of the Army position unless so designated by other authorized documents. Comments or suggestions should be addressed to:

Commander
USA Concepts Analysis Agency
ATTN: Director of Methodology and Computer Support
8120 Woodmont Avenue
Bethesda, Maryland 20014

CAA-D-80-1

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) CAA-D-80-1	2. GOVT ACCESSION NO. AD-A091	3. RECIPIENT'S CATALOG NUMBER 653
4. TITLE (and Subtitle) (6) PROGTEST: A Computer System for the Analysis of Computational Computer Programs		5. TYPE OF REPORT & PERIOD COVERED (9) DOCUMENTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Dr. Steve Bravy		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Concepts Analysis Agency 8120 Woodmont Avenue Bethesda, Maryland 20014		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Methodology and Computer Support Directorate US Army Concepts Analysis Agency 8120 Woodmont Avenue, Bethesda, MD 20014	(11)	12. REPORT DATE April 1980
		13. NUMBER OF PAGES 130
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 125		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer debugging, computer benchmarking, computer program, computer routine testing package, computer aided diagnosis, computer program reliability, computer program verification, software diagnosis, software testing, software reliability, software verification, model testing, model diagnosis, model reliability, model verification.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This system assists the testing and debugging of computational computer programs by generating test points for the tested program at each test point. This information may be used to infer the operation of, or to predict output values of an operational program as well. The performance data highlighting input variable sensitivities is analyzed further by one of the components of the system.

(Over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

390776

20. Abstract (continued)

This writeup contains complete documentation of the system, including description of the methodology, sample outputs, flowcharts and listings.

The package is written in UNIVAC's FORTRAN V.

DOCUMENTATION
CAA-D-80-1

PROGTEST: A COMPUTER SYSTEM FOR THE ANALYSIS OF COMPUTATIONAL COMPUTER PROGRAMS

April 1980

Prepared by

Methodology and Computer Support Directorate

US Army Concepts Analysis Agency
8120 Woodmont Avenue
Bethesda, Maryland 20014

ABSTRACT

1. PROGTEST. This Computer System for the Analysis of Computational Computer Programs consists of three interrelated subsystems:

POINTCOMP	consisting of the major components POINTCOMP and VARVARY1
ANALYZ	containing the ANALYZ routine
GRID	consisting of the GRID, REARRANGE and DIFFQUOT routines as major components

2. POINTCOMP

a. The POINTCOMP routine of the POINTCOMP subsystem utilizes the method of steepest ascent/descent to generate input variable values maximizing/minimizing the output of the program being tested (one output at a time). The points generated by the routine are printed out and may be used by the analyst to determine whether increasing or decreasing the variable increases/decreases the output, or whether a maximum/minimum is reached. The routine also prints out sensitivity information usable by the analyst in determining the comparative effect of each variable upon the output variable. Points generated by this routine are used as input by some of the other subsystems.

b. The VARVARY1 component of the POINTCOMP subsystem uses input variable values produced by POINTCOMP or chosen by the analyst. For each combination produced or chosen by the analyst, each variable is varied uniformly through a range while keeping the other variables fixed. The tested routine is evaluated at each new point and the gradient is computed at each new point. In addition, statistics showing the average marginal return over various portions of the variables range are printed out. The output of this program can be used to generate curves showing the effect of the variable being varied upon the output.

3. ANALYZ. The ANALYZ subsystem uses a file of gradients as input. This type of file is produced by several subsystems. The subsystem computes and outputs sensitivity statistics derived from the gradients.

4. GRID. The GRID routine of the GRID subsystem utilizes a user defined grid to intensively analyze a small area of input values. The subsystem evaluates the routine to be tested at each node in the grid, and prints out the input values, the evaluated values, and the gradients at each point. The REARRANGE routine of the GRID subsystem rearranges the data to show the variable by variable variation, and the DIFFQUOT routine computes difference quotients for the generated points for one variable at a time.

CONTENTS

PROGTEST: A COMPUTER SYSTEM FOR THE ANALYSIS OF COMPUTATIONAL COMPUTER PROGRAMS

CHAPTER 1

OUTLINE OF SUBSYSTEMS

1-1. SUBSYSTEM DESCRIPTIONS

a. POINTCOMP Subsystem

(1) POINTCOMP. The POINTCOMP routine of the POINTCOMP subsystem creates input variable values by using maximum ascent/descent techniques to maximize/minimize the output of the program being tested. The output from POINTCOMP includes:

(a) A file in standard format for which data lines are comprised of variable values and the corresponding value of the program being tested. This file is created on FORTRAN unit 10 and is one possible input to VARVARY1. FORTRAN allows usage of a disk file as on I/O unit.

(b) A file of the corresponding gradients in standard format on FORTRAN unit 11.

(c) A listing containing the new points and the corresponding program values, the gradients, and the computed increments is produced on the printer.

(d) The subsystem includes the following routines:

1. POINTCOMP.

2. PARTL.

3. PREPR.

4. The program to be tested.

(2) VARVARY1. The VARVARY1 component of the POINTCOMP subsystem takes the file created by POINTCOMP on unit 10 as input or a similarly structured user provided file. The user must also input a file containing higher and lower bounds and increments for each variable. VARVARY1 will read in each line of the unit 10 file and from each point inputted VARVARY1 creates new points by varying each variable in turn through the given range using the

given increment keeping the other values fixed. For each new point, the program to be tested is evaluated and the point and the corresponding tested program value are printed out.

(3) In addition, The VARVARY1 routine computes the gradient at each new point and prints its components. The new points and the corresponding tested routine values are written out on unit 12 and the corresponding gradients are written out on unit 11. When one variable is varied and the others held constant, the difference in the output values divided by the difference in the varied variable values is called the difference quotient. As the VARVARY1 routine varies a variable through its range, various combinations of difference quotients are computed and printed for that variable.

(4) The output from the VARVARY1 routine includes:

(a) A listing of the new input points and the associated output values of the routine being tested. This listing also contains the difference quotients.

(b) A listing of the corresponding gradients.

(c) A file containing the new gradients in standard format on unit 11.

(d) A file containing the new generated points in standard format on unit 12.

(e) A scratch file containing input variable values and the associated output values on unit 14.

(f) The subsystem includes the following routines:

1. VARVARY1.

2. PARTL.

3. PREPR.

4. The program being tested.

b. ANALYZ Subsystem. This subsystem analyzes a file of gradients in standard format on unit 11 produced by the POINTCOMP, VARVARY1, or GRID subsystem. The statistics produced include:

(1) For each gradient, the ratio of every component to the minimum component in absolute value.

(2) For each component of each gradient the change needed in the corresponding variable in order to change the output by one unit.

(3) For each component of each gradient:

(a) If the component is not the largest in absolute value, the ratio of the component to the absolute value of the maximal component.

(b) For the maximal component, the ratio of the absolute value of the maximal component to the absolute value of the next largest components.

(4) The gradients are also aggregated component by component, and statistics (1) to (3) described above are computed for the aggregate vector.

(5) The output from ANALYZ includes a listing of the statistics described above.

(6) The subsystem includes the following routine: ANALYZ.

c. GRID Subsystem. This subsystem is comprised of three linked but independent components--GRID, REARRANGE, and DIFFQUOT.

(1) GRID. The GRID subsystem needs a range and step size for each variable. The subsystem varies each variable from the lower bound to the upper bound, incrementing each value by a given step size to obtain the next value., thereby producing all combinations of variable values. The routine to be tested is evaluated at each point and the gradient is computed at each point as well.

(a) The outputs from GRID include:

1. A listing of the new points and the associated routine values.

2. A listing of the gradients.

3. A file in standard format containing the new points and the associated values on unit 10. This file is one possible input to REARRANGE.

4. A file in standard format containing the corresponding gradients on unit 11. This file is the other possible input to REARRANGE.

5. A scratch file containing the new points, the associated values and some pointers on unit 12. This file is used as an input to REARRANGE.

6. A scratch file containing the gradients and tested routine values on unit 13.

(b) The components of this subsystem are:

1. GRID.

2. PARTL.

3. PREPR.

4. The program to be tested.

(2) REARRANGE. This routine reads from unit 14, one of two files produced by GRID on units 12 or 13. The routine also utilizes the file produced by GRID on unit 12. The output data is rearranged variable by variable to facilitate further analyses.

(a) The output from REARRANGE includes:

1. A listing of the rearranged input file, where adjoining lines in the same section differ only in one variable value, each point is numbered to facilitate linkage of this output to the GRID output.

2. A file containing similar information produced on unit 15.

(b) The subsystem has one component: REARRANGE.

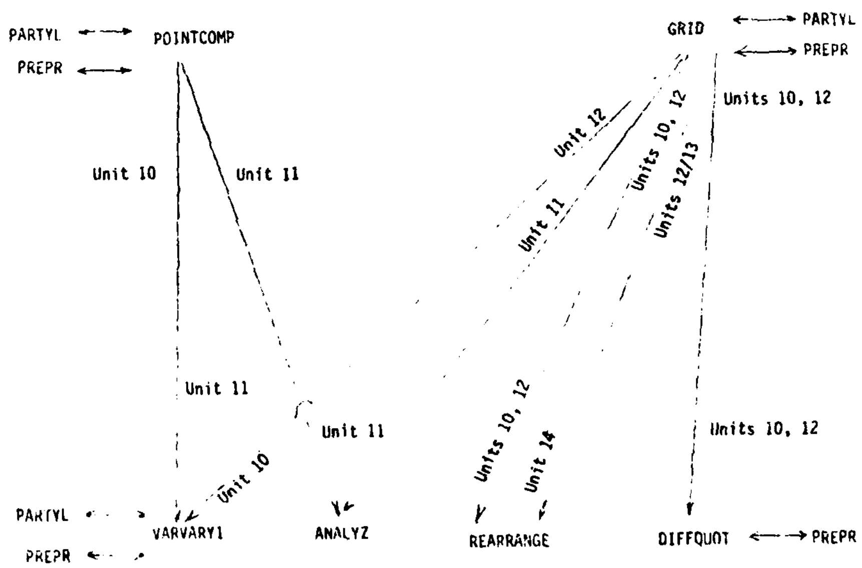
(3) DIFFQUOT. This routine reads the GRID output files produced on units 10 and 12. The routine outputs a list of difference quotients for the variable requested. The outputted quotients are headed by the numbers of the relevant GRID points, facilitating the linkage between the GRID output and this output. The subsystem has one component: DIFFQUOT.

1-2. DESCRIPTION OF SUBSYSTEMS. Subsystem components are shown in Table 1-1.

Table 1-1. Subsystem Composition

Subsystem	Major components	Minor components
POINTCOMP	POINTCOMP VARVARY1	PARTL PREPR
ANALYZ	ANALYZ	--
GRID	GRID REARRANGE DIFFQUOT	PARTL PREPR

1-3. SUBSYSTEM INPUTS AND OUTPUTS. Figure 1-1 and Table 1-2 show inputs, outputs, and intercommunication between the subsystems.



NOTES: POINTCOMP, VARVARY1, and GRID routines each may be run independently of the other major routines. The other major routines may be run independently once POINTCOMP or GRID have been run to get up the required input files, as indicated above.

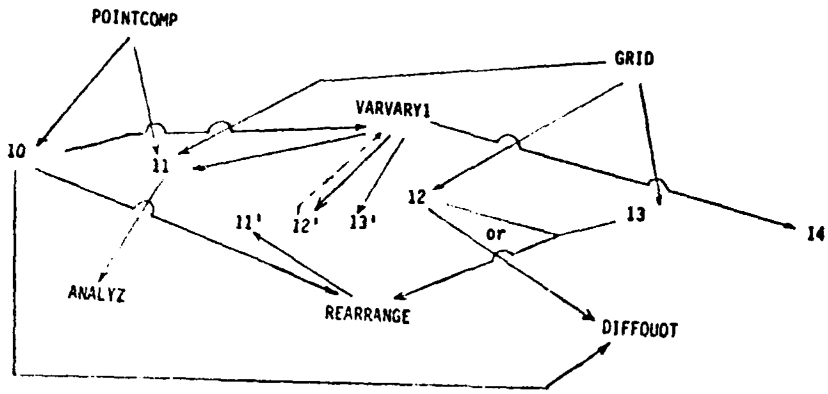
The head of the arrow denotes input; the tail of the arrow, output.

Figure 1-1. Data Flow

Table 1-2. Unit Utilization (all routines utilize Units 5 and 6 in addition to those listed below.)

Routine name	Unit number	Type	Format	Description
POINTCOMP	10	Output	Standard	Each line contains input variable values and the corresponding value.
	11	Output	Standard	Each line contains the corresponding gradients.
VARVARY1	10	Input	Standard	File created by PONTCOMP, containing the the variable values.
	11	Output	Standard	Gradients corresponding to new points génerated by VARVARY1.
	12	Output	Standard	New points generated by VARVARY1 and the associated value of the tested routine.
	13	Output	Nonstandard	Scratch file containing gradients.
	14	Output	Nonstandard	Scratch file containing input variable values.
ANALYZ	11	Input	Standard	Gradient file created by POINTCOMP, VARVARY1, or GRID
GRID	10	Output	Standard	File containing input variable values and the associated values of the tested routine.
	11	Output	Standard	File containing the corresponding gradients.
	12	Output	Nonstandard	Scratch file containing the input variable values tested routine value and a pointer to the variable varied to create the point. Used as input to REARRANGE.
	13	Output	Nonstandard	Scratch file containing the corresponding gradients and the value of the tested routine. Used as input to REARRANGE.
REARRANGE	10	Input	Standard	File produced by GRID, containing the input variable values and tested routine value on Unit 12, containing the indication of which variables were varied.
	12	Input	Nonstandard	File produced by GRID on Unit 12, containing the indication of which variable was varied. Used in the reordering computations by REARRANGE.
	14	Input	Nonstandard	The file to be reordered, this file could be the file produced by GRID on Units 12 or 13.
	15	Output	Nonstandard	A file containing the rearranged output.
DIFFQUOT	10	Input	Standard	File containing input variable values, produced by GRID.
	12	Input	Nonstandard	File produced by GRID, indicating which variables were varied to create each point.

The following (Figure 1-2) outlines the units utilized by the different routines.



NOTE: Numbers denote I/O unit numbers; primes denote same unit but different formats.

Figure 1-2. Subsystem Unit Utilization

1-4. STANDARD FILE STRUCTURE. A standard file has the following structure:

a. Line 1 is Comprised of Fields A1, A2, and A3

(1) A1 is the number of points or gradients in I5 format.

(2) A2 is the number of variables in I3 format.

(3) A3 is a positive number in F10.7 format. Any vector components smaller than this number in absolute value will be considered to be zero.

b. Line 2 (format). This line indicates the format that was used to write each data line in the file and which may be used to read the subsequent lines in the file.

c. Other Lines. These lines contain data consisting either of variable value vectors (points) or gradients. The leftmost column gives data on variable 1, the next column to the right on variable 2, etc.

CHAPTER 2

THE POINTCOMP SUBSYSTEM

Section I. THE POINTCOMP ROUTINE

2-1. INTRODUCTION. This routine uses the gradient to pick new points which minimize/maximize (locally) the value of the output of the routine being tested. In addition, the gradient at each point is also printed out. The analyst may use the output to check out the routine being tested in the following manner:

a. The analyst can examine each variable to see if it must be increased/decreased to increase/decrease the output. The analyst can also check for attainment of values from which no increase/decrease of the output variable is possible. These results should be explainable intuitively.

b. The analyst may examine each gradient to determine the relative effect of each variable at that point.

c. The analyst may examine the components of the gradients for all points to analyze the marginal returns.

d. Changes in the sign of a component of a gradient from one point to the next indicate that a potential local optimum for that variable lies between the two successive values of the coordinate.

e. POINTCOMP calls PARTL to compute partial derivatives numerically. POINTCOMP calls PREPR, a user provided driver routine which obtains an output value from the program being tested.

2-2. BACKGROUND

If $f(x_1, \ldots, x_n)$ is a real valued function of several variables,

$$\nabla f(x_1, \ldots, x_n) = \left(\frac{\partial f}{\partial x_1}(x_1, \ldots, x_n), \ldots, \frac{\partial f}{\partial x_n}(x_1, \ldots, x_n)\right)$$

points in the direction of greatest local increase in f from $(x_1, \ldots, x_n)$, and $-\nabla f(x_1, \ldots, x_n)$ points in the direction of greatest local decrease in f from $(x_1, \ldots, x_n)$. We justify this fact by the following argument:

If $(y_1, \ldots, y_n)$ is a point "close" to $(x_1, \ldots, x_n)$, $f(y_1, \ldots, y_n) - f(x_1, \ldots, x_n)$ is approximated by:

$$\sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x_1, \ldots, x_n)\,(y_i - x_i)$$
$$= \nabla f(x_1, \ldots, x_n) \cdot (y - x)$$
$$= ||\nabla f(x_1, \ldots, x_n)||\; ||y - x||\cos\theta$$

where: $y = (y_1, \ldots, y_n)$;

$x = (x_1, \ldots, x_n)$;

$||A||$ is the magnitude of a vector A; and

θ is the angle between $\nabla f(x_1, \ldots, x_n)$ and $(y - x)$.

So

$$f(y_1, \ldots, y_n) - f(x_1, \ldots, x_n) = ||\nabla f(x_1, \ldots, x_n)||\; ||y - x||\cos\theta,$$

and therefore:

$$\frac{f(y_1, \ldots, y_n) - f(x_1, \ldots, x_n)}{||y - x||} = ||\nabla f(x_1, \ldots, x_n)||\cos\theta$$

Now $-1 \leq \cos\theta \leq 1$, so

$$\max_{||y-x||\neq 0} \frac{f(y_1 \ldots, y_n) - f(x_1 \ldots, x_n)}{||y - x||} = ||\nabla f(x_1 \ldots, x_n)||.$$

Therefore the greatest change in the function f per unit distance between y and x (i.e., $||y - x||$) is the magnitude of the gradient $||\nabla f(x_1, \ldots, x_n)||$. This maximum is reached for $\cos\theta = 1$ or $\theta = 0$, so the direction of greatest increase is $\theta = 0$, so $y - x$ and $f(x_1, \ldots, x_n)$ are collinear. Therefore, $\nabla f(x_1, \ldots, x_n)$ gives both the magnitude and direction of the maximal change.

Likewise:

$$\min_{||y-x||\neq 0} \frac{f(y_1 \ldots, y_n) - f(x_1 \ldots, x_n)}{||y - x||} = -||\nabla f(x_1 \ldots, x_n)||$$

In this case, $\cos \theta = -1$, so $\theta = 180°$ and the direction of greatest decrease (least increase) per unit distance between y and x ($||y - x||$) is $-\nabla f(x_1, \ldots, x_n)$, the magnitude of greatest decrease is $||\nabla f(x_1, \ldots, x_n)||$.

2-3. DISCUSSION OF METHODOLOGY. Several criteria are used in the program to decide upon the distance to be moved along the gradient in order to obtain the next point. These criteria are:

a. The maximum change in one step for each variable (an input parameter) divided by the size of the component of the gradient corresponding to that variable.

b. First, compute a certain fraction of the difference between the current variable value and the bound on the variable in the appropriate direction (an input parameter) whenever the bound exists. When the variable isn't bounded, use twice the current value as a bound. Next, the number just described is divided by the size of the corresponding component of the gradient.

c. Compute the minimum of the numbers developed in parts a and b and the default multiplier read in as an input parameter on the 8th input card (see pg 2-6). This step computes a gradient multiplier.

d. A candidate increment is determined by multiplying the gradient by the number derived by the process described in c. If the gradient multiplier is too small, use the A7 field on card 1. If necessary, the increment is multiplied by a number sufficient to increase each component to the minimum threshold. These minimum thresholds are also input.

e. Lastly, the increment vector is added to the original point, obtaining a new point candidate. The components of this new point exceeding the bounds of the corresponding variable (if any) are set equal to the value of the bound, and the result is taken to be the next point.

2-4. LIMITATIONS. As currently compiled, the POINTCOMP routine is subject to the following limitations:

a. Testing of the program must proceed by testing one output variable at a time if the program has several output variables.

b. If any variables input to the program being tested are integers, it is possible that their incrementation and effect upon the output variables will be reduced due to truncation.

c. Current limits are set to 20 variables and 500 generated points at most. This is easily changed.

d. Testing Monte Carlo programs is feasible, but potentially time consuming. The PREPR routine, called by POINTCOMP, must call the routine being tested repetitively and average the output values before returning. Replication is necessary in order to average out random effects. A better approach is to use the Monte Carlo variables as ordinary input variables and let POINTCOMP assign their values with no replication.

e. The function represented by the routine to be tested should be well-behaved (i.e., differentiable).

f. The order in which the criteria for step size are examined is fixed, so there is a hard wired priority among the criteria.

2-5. RUN SETUPS

a. Developing an Absolute File in ASCII

```
@MAP,S          ,Name of absolute element
IN              03PROGTEST. POINTCOMP
IN              03PROGTEST. PARTIAL
IN              Element containing PREPR subroutine
IN              Programs to be tested
LIB             LIB$*FTN3.
END
```

b. To Execute

```
@USE        10, file name to contain variable values and
            values of the tested routine
@USE        11, data file name for gradient file in standard format
@ASG,A      First file name
@ASG,A      Second file name
@XQT        Absolute element created by the @MAP
Input
Deck
```

c. Input Deck. The following tables describe the input deck (Table 2-1), show a sample input deck (Table 2-2), and a sample run deck (Table 2-3).

Table 2-1. Description of Input Deck
(page 1 of 2 pages)

Line 1

Field:	A1	A2	A3	A4	A5	A6	A7	A8
Format:	I3	I3	I2	F10.7	F10.7	F12.0	F12.5	I3

where the fields are defined as follows:

A1 is the maximum number of points to be generated.

A2 is the number of variables.

A3 is +1 if output is to be increased, -1 if output is to be decreased.

A4 is a positive real number. Any number smaller than this number will be considered to be zero.

A5 is a positive real number, gradients smaller than this number will be considered to be zero.

A6 is a code number. This number can be used to indicate that a variable is unbounded above or below.

A7 describes the minimal multiplier desired for the gradient.

A8 the run is executed in a debugging mode (more printout) when this field contains a -1.

Line 2. (Format for reading in each of the following six lines, lines 3-8 which follow. Parentheses must bound the format as shown.)

Line 3. Initial values, one for each variable, in the format described by line 2.

Table 2-3. Description of Input Deck
(page 2 of 2 pages)

Line 4. Lower bounds allowed for each variable, in the format described by line 2. If some variable is unbounded below, indicate this by using the number from field A6 of the first line.

Line 5. Upper bounds allowed for each variable. Rest of description as in line 4.

Line 6. The preferred maximum single step change in each variable, one number for each variable.

Line 7. The preferred minimum single step change in each variable, one for each variable.

Line 8. Default gradient multiplier for each variable, one number for each variable. These will only be used for unbounded variables and are only useful to reduce the number by which the gradient will be multiplied.

Line 9. (Format for writing one line of variable values and the corresponding program output value on unit 10. This format should accommodate at least n+3 values, where n is the number of variables.)

Line 10. (Format for writing out one line of variable values and the corresponding program output value on the printer.)

Table 2-2. Sample Input Data

4[a] 9[b]+1[c]	.001[d]	.001[e]	1001.[f]	.1000[g]+1[h]

(13F5.0) format for inputting the following six lines

5.0	0.5	0.5	0.5	Initial value
0.0	0.0	0.0	0.0	Lower bounds
1000.0	1000.0	1000.0	0.9	Upper bounds
10.0	10.0	10.0	0.5	Preferred maximum step
0.0	0.0	0.0	0.0	Preferred minimum step
1.0	1.0	1.0	1.0	Default multiplier (not really needed for bounded variables)

First column describes variable 1.

Second column describes variable 2.

Etc.

(6(7X,F13.5))	Format for writing values and output on unit 10
(1X,6(fX,F13.5))	Format for writing values and output on the printer

[a]Number of variables.

[b]Maximum number of points.

[c]Increase the output.

[d]Numbers lower than this are considered zero.

[e]Gradients smaller than this are considered zero.

[f]Code number indicating no bounds.

[g]Minimal gradient multiplier.

[h]Not in debug mode.

Table 2-3. Sample Run Deck

```
@USE 10,03MAT1.

@ASG,A 03MAT1.

@USE 11,03MAT2.

@ASG,A 03MAT2.

@XQT absolute deck created by @MAP

ØØ4ØØ9 +1          .001         .001        1001.      .1000 +1
(13F6.0)
   5.0             0.5          0.5          0.5
   0.0             0.0          0.0          0.0
1000.0          1000.0       1000.0          0.9
  10.0            10.0         10.0          0.5
   0.0             0.0          0.0          0.0
   1.0             1.0          1.0          1.01
(6(7X,F13.5))
(1X,6(7X,F13.5))
```

2-6. OUTPUT DESCRIPTIONS AND SAMPLE OUTPUT

a. Tuples. Each row corresponds to a point where the rightmost column is the value of the routine being tested and the leftmost column represents variable 1, the column to its right variable 2, etc.

b. Gradients. Each row is the gradient at the point described by the corresponding row of the tuples output, e.g., the first row is the gradient at the point described by the first row of the tuples printout.

c. Increments. Each row is the computed vector difference of the corresponding tuples (except where boundaries are encountered). The first row is the second tuple minus the first, the second row is the third tuple minus the second, etc. The following (Table 2-4) exhibits sample printer output.

Table 2-4. Sample Output to the Printer

Tuples				
5.00000	.50000	.50000	.50000	.73821
5.00000	.61902	.75809	.63333	.99916
5.00119	.73820	.97516	.76667	1.24159
5.00270	.80924	1.08781	.84667	1.37902
5.00270	.84061	1.13401	.88222	1.43833
5.00270	.85178	1.14995	.89492	1.45926
5.00270	.85513	1.15468	.89873	1.46551
5.00270	.85513	1.15590	.89873	1.46626
5.00270	.85513	1.15715	.89975	1.46753
Gradients				
.00048	.30919	.67048	.34638	
.00359	.35940	.65458	.40207	
.00852	.40086	.63565	.45144	
.01236	.42295	.62294	.47942	
.01428	.43221	.61698	.49139	
.01500	.43545	.61481	.49560	
.01522	.43641	.61415	.49686	
.01523	.43631	.61415	.49674	
.01527	.43670	.61405	.49663	
Increments				
.00000	.11902	.25809	.13333	
.00119	.11919	.21707	.13333	
.00151	.07104	.11264	.08000	
.00000	.03137	.04620	.03556	
.00000	.01117	.01594	.01270	
.00000	.00335	.00473	.00381	
.00000	.00000	.00122	.00000	
.00000	.00000	.00126	.00102	

d. Output File Descriptions

(1) File on Unit 10

(a) Line 1

Fields	A1	A2	A3
Formats	I5	I3	F10.7

where

A1 is the number of points in the file.

A2 is the number of variables.

A3 is the number determining when small numbers are considered to be zero.

(b) Line 2. (Format for reading in one point and its corresponding program value.)

(c) Other Lines. Each line contains a set of variable values and the corresponding tested program value.

(2) File on Unit 11

(a) Same as file on unit 10.

(b) Same as file on unit 10.

(c) Other lines - Each line contains the corresponding gradient.

e. Sample Output File--Unit 10

```
  9  4  .0010000
(6(7X,F13,5))
```

Tuples

5.00000	.50000	.50000	.50000	.73821
5.00000	.61902	.75809	.63333	.99916
5.00119	.73820	.97516	.76667	1.24159
5.00270	.80924	1.08781	.84667	1.37902
5.00270	.84061	1.13401	.88222	1.43833
5.00270	.85178	1.14995	.89492	1.45926
5.00270	.85513	1.15468	.89873	1.46551
5.00270	.85513	1.15590	.89873	1.46626
5.00270	.85513	1.15715	.89975	1.46753

f. Sample Output File--Unit 11

```
  9  4  .0010000
(6(7X,F13,5))
```

.00048	.30919	.67048	.34638
.00359	.35940	.65458	.40207
.00852	.40086	.63565	.45144
.01236	.42295	.62294	.47942
.01428	.43221	.61698	.49139
.01500	.43545	.61481	.49560
.01522	.43641	.61415	.49686
.01523	.43631	.61415	.49674
.01527	.43670	.61405	.49663

2-7. POINTCOMP ROUTINE LISTING

```
      PARAMETER FILE1=10,FILE2=11,NOVALS=20,NOPTS=500,INFILE=5,OUTFIL=6
      PARAMETER NOVAL1=NOVALS+1
C
      CHARACTER*80 FORMT1,FORMT2,FORMT3
C
      DIMENSION FSTVAL(NOVALS),LOWBDS(NOVALS),HYBDS(NOVALS)
      DIMENSION MAXCHG(NOVALS),GRAD(NOPTS,NOVALS),TUPLES(NOPTS,NOVAL1)
      DIMENSION DELTA1(NOVALS),DELTA2(NOVALS),DELTA(NOPTS,NOVALS)
      DIMENSION SIGN(NOVALS),INPUT(NOVALS),MIN(NOVALS),NOMULT(NOVALS)
C
      INTEGER NOVARS,UPDOWN,MAXPTS,SIGN,I,J,P,Q,SWITCH
C
      REAL EPSLON,FLAT,NOLIM,FSTVAL,LOWBDS,HYBDS,MAXCHG
      REAL GRAD,SUM,INPUT,INCR,PARTYL,DELTA1,DELTA2,DELTA
      REAL TUPLES,VALUE,NOMULT,MIN,MIN1,MINMLT,MAX
C
      REWIND FILE1
      REWIND FILE2
      READ(INFILE,10000) NOVARS,MAXPTS,UPDOWN,EPSLON,FLAT,NOLIM,
     1  MINMLT,SWITCH
10000 FORMAT(2I3,I2,2F10.7,2F12.0,I3)
C *****
      IF (SWITCH .NE. -1) GOTO 30010
      WRITE(OUTFIL,20060) NOVARS,MAXPTS,UPDOWN,EPSLON,FLAT,NOLIM,MINMLT
20060 FORMAT(' **INVALS**',/,1X,2I3,I2,2F10.7,F12.0,F12.5)
30010 CONTINUE
C *****
      READ(INFILE,10010) FORMT1
10010 FORMAT(A80)
      READ(INFILE,FORMT1) (FSTVAL(I),I=1,NOVARS)
      READ(INFILE,FORMT1) (LOWBDS(I),I=1,NOVARS)
      READ(INFILE,FORMT1) (HYBDS(I),I=1,NOVARS)
      READ(INFILE,FORMT1) (MAXCHG(I),I=1,NOVARS)
      READ(INFILE,FORMT1) (MIN(I),I=1,NOVARS)
      READ(INFILE,FORMT1) (NOMULT(I),I=1,NOVARS)
      READ(INFILE,10010) FORMT2
      READ(INFILE,10010) FORMT3
C *****
      IF (SWITCH .NE. -1) GOTO 30020
      WRITE(OUTFIL,21000) (FSTVAL(I),I=1,NOVARS)
21000 FORMAT(' **FSTVAL**',10(/,1X,6F20.10))
      WRITE(OUTFIL,21010) (LOWBDS(I),I=1,NOVARS)
21010 FORMAT(' **LOWBDS**',10(/,1X,6F20.10))
      WRITE(OUTFIL,21020) (HYBDS(I),I=1,NOVARS)
21020 FORMAT(' **HYBDS**',10(/,1X,6F20.10))
      WRITE(OUTFIL,21030) (MAXCHG(I),I=1,NOVARS)
21030 FORMAT(' **MAXCHG**',10(/,1X,6F20.10))
      WRITE(OUTFIL,20065) (MIN(I),I=1,NOVARS)
20065 FORMAT(' **MINS**',10(/,1X,6F20.10))
      WRITE(OUTFIL,20067) (NOMULT(I),I=1,NOVARS)
20067 FORMAT(' **NOMULT**',10(/,1X,6F20.10))
30020 CONTINUE
C *****
```

```
C       INITIALIZATION
      DO 100 I=1,NOVARS
      TUPLES(1,I)=FSTVAL(I)
      INPUT(I)=FSTVAL(I)
100   CONTINUE
      CALL PREPR(INPUT,VALUE)
      TUPLES(1,NOVAL1)=VALUE
C       LOOP TO CREATE POINTS
      DO 3000 P=2,MAXPTS
      DO 300 I=1,NOVARS
      INPUT(I)=TUPLES(P-1,I)
300   CONTINUE
      DO 400 I=1,NOVARS
      CALL PARTL(NOVARS,I,INPUT,EPSLON,PARTYL,INCR,TUPLES(P-1,NOVAL1))
C *****
      IF (SWITCH .NE. -1) GOTO 30030
      WRITE(OUTFIL,20000) (INPUT(J),J=1,NOVARS),PARTYL,INCR,
     1   TUPLES(P-1,NOVAL1)
20000 FORMAT(//,' **INPUT PARTIAL**',10(/,1X,6F20.10))
30030 CONTINUE
C *****
      GRAD(P-1,I)=PARTYL
400   CONTINUE
      DO 500 I=1,NOVARS
      SIGN(I)=UPDOWN
      IF (GRAD(P-1,I) .LT. 0.) SIGN(I)=-UPDOWN
500   CONTINUE
C *****
      IF (SWITCH .NE. -1) GOTO 30040
      WRITE(OUTFIL,20010) (GRAD(P-1,I),I=1,NOVARS)
20010 FORMAT(//,' **GRAD**',10(/,1X,6F20.10))
      WRITE(OUTFIL,20015) (SIGN(I),I=1,NOVARS)
20015 FORMAT(//,' **SIGN**',10(/,1X,10I3))
30040 CONTINUE
C *****
C COMPUTE CANDIDATE MULTIPLIER BASED MAX CHANGE PER COORDINATE
      DO 600 I=1,NOVARS
      DELTA1(I)=ABS(NOMULT(I))
      IF (ABS(GRAD(P-1,I)) .LT. EPSLON) GOTO 600
      DELTA1(I)=ABS(MAXCHG(I))/ABS(GRAD(P-1,I))
600   CONTINUE
C *****
      IF (SWITCH .NE. -1) GOTO 30050
      WRITE(OUTFIL,20020) (DELTA1(I),I=1,NOVARS)
20020 FORMAT(//,' **DELTA1**',10(/,1X,6F20.10))
30050 CONTINUE
```

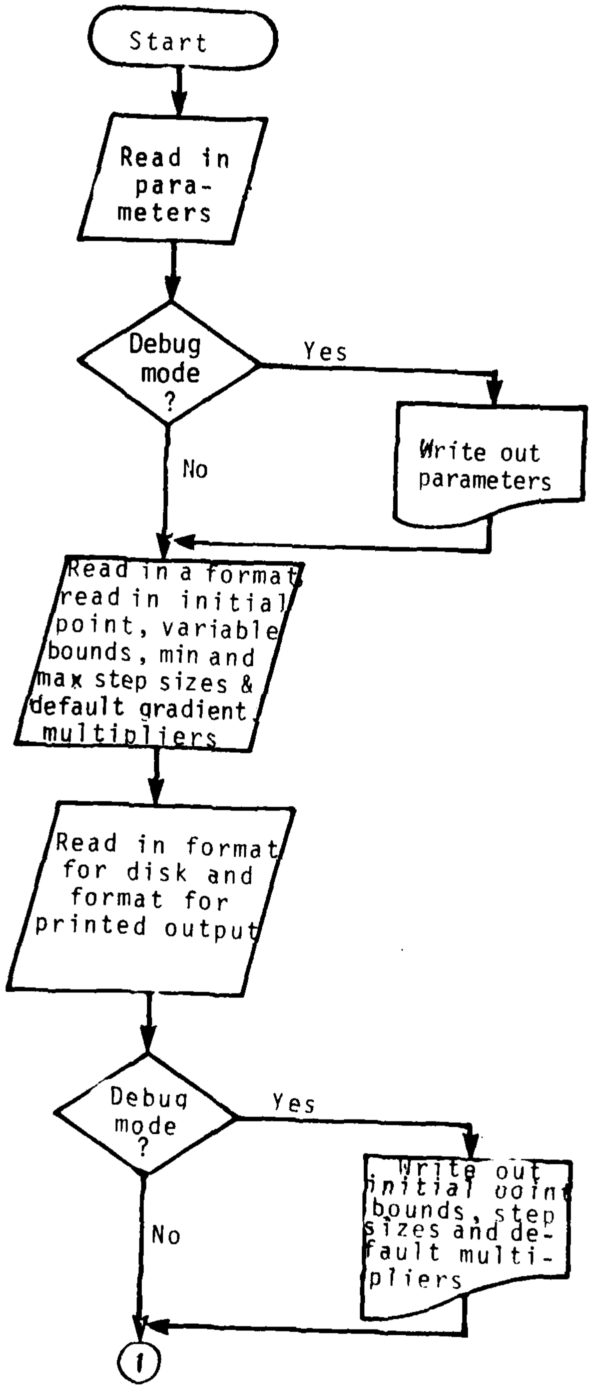
```
C *****
C COMPUTE CANDIDATE MULTIPLIER BASED ON DISTANCE FROM BOUNDS
C      OR DOUBLING THE VALUE IF NO BOUND EXISTS
       DO 800 I=1,NOVARS
       DELTA2(I)=AMIN1(ABS(TUPLES(P-1,I)/ABS(GRAD(P-1,I))),NOMULT(I))
       IF (SIGN(I) .LT. 0 ) GOTO 700
       IF (ABS(HYBDS(I)-NOLIM) .GE. EPSLON) DELTA2(I)=
      1 ((FLOAT(P)-1.)/(FLOAT(P)+1.))*(ABS(HYBDS(I)-TUPLES(P-1,I)))
      2   /ABS(GRAD(P-1,I))
       GOTO 800
700    IF (ABS(LOWBDS(I)-NOLIM) .GE. EPSLON) DELTA2(I)=
      1 ((FLOAT(P)-1.)/(FLOAT(P)+1.))*(ABS(LOWBDS(I)-TUPLES(P-1,I)))
      2   /ABS(GRAD(P-1,I))
800    CONTINUE
C *****
       IF (SWITCH .NE. -1) GOTO 30060
       WRITE(OUTFIL,20046) (DELTA2(I),I=1,NOVARS)
20046 FORMAT(//,' **DELTA2**',10(/,1X,4F30.10))
30060 CONTINUE
C *****
C      COMPUTE THE INCREMENTATION VECTOR
       DO 900 I=1,NOVARS
       DELTA(P-1,I)=AMIN1(DELTA1(I),DELTA2(I))
900    CONTINUE
C *****
       IF (SWITCH .NE. -1) GOTO 30070
       WRITE(OUTFIL,20045) (DELTA(P-1,I),I=1,NOVARS)
20045 FORMAT(//,' **DELTA.0**',10(/,1X,6F20.10))
30070 CONTINUE
C *****
C TO COMPUTE THE MINIMUM GRADIENT MULTIPLIER
       MIN1=DELTA(P-1,1)
       DO 925 I=1,NOVARS
       IF (DELTA(P-1,I) .LT. EPSLON) GOTO 925
       IF (DELTA(P-1,I) .LT. MIN1) MIN1=DELTA(P-1,I)
925    CONTINUE
C *****
       IF (SWITCH .NE. -1) GOTO 30080
       WRITE(OUTFIL,20044) (DELTA(P-1,I),I=1,NOVARS),MIN1
20044 FORMAT(//,' **DELTA.1**',10(/,1X,6F20.10))
30080 CONTINUE
C*****
C TO COMPUTE A CANDIDATE INCREMENT
       IF (MIN1 .LT. EPSLON) MIN1=MINMLT
       DO 950 I=1,NOVARS
       DELTA(P-1,I)=MIN1*ABS(GRAD(P-1,I))
950    CONTINUE
C *****
```

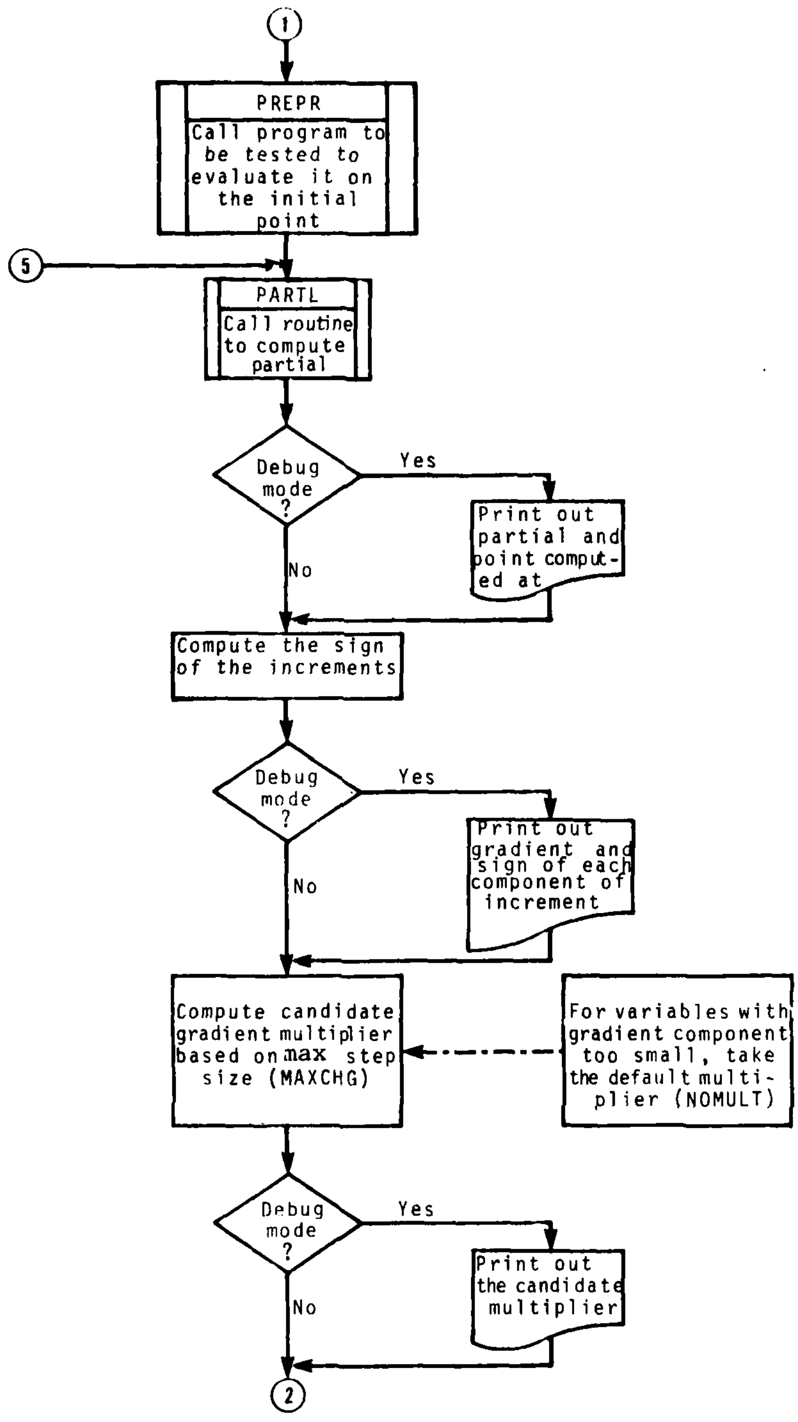
```
      IF (SWITCH .NE. -1) GOTO 30090
      WRITE(OUTFIL,20040) (DELTA(P-1,I),I=1,NOVARS),MIN1
20040 FORMAT(//,' **DELTA.2,MIN1**',10(/,1X,6F20.10))
30090 CONTINUE
C *****
C TO GUARANTEE A MINIMAL STEP IN EACH COORDINATE
      MAX=0.
      DO 990 I=1,NOVARS
C SET MINISCULE STEPS TO ZERO
      IF (DELTA(P-1,I) .GE. EPSLON) GOTO 980
      DELTA(P-1,I)=0.
      GOTO 990
C CHECK TO SEE IF CANDIDATE INCREMENT EXCEEDS MINIMUM
980   IF ( DELTA(P-1,I) .GE. MIN(I) ) GOTO 990
C *****
      IF (SWITCH .NE. -1) GOTO 30200
      WRITE(OUTFIL,20200) I,MAX,MIN(I)/ABS(DELTA(P-1,I))
20200 FORMAT(/,100(/,' **MAX**',I5,4F20.10))
30200 CONTINUE
C *****
      IF ((MIN(I)/DELTA(P-1,I)) .LE. MAX) GOTO 990
      MAX=MIN(I)/DELTA(P-1,I)
990   CONTINUE
      IF ( MAX .LT. EPSLON) GOTO 975
C INCREASE THE STEP SIZE
      DO 970 I=1,NOVARS
      DELTA(P-1,I)=MAX*DELTA(P-1,I)
970   CONTINUE
C *****
      IF (SWITCH .NE. -1) GOTO 30110
      WRITE(OUTFIL,20042) (DELTA(P-1,I),I=1,NOVARS),MAX
20042 FORMAT(//,' **DELTA.3,MAX**',10(/,1X,6F20.10))
30110 CONTINUE
C *****
975   DO 985 I=1,NOVARS
      IF ( DELTA(P-1,I) .GT. MAXCHG(I) ) DELTA(P-1,I)=MAXCHG(I)
      DELTA(P-1,I)=DELTA(P-1,I)*SIGN(I)
985   CONTINUE
C *****
      IF (SWITCH .NE. -1) GOTO 30120
      WRITE(OUTFIL,20043) (DELTA(P-1,I),I=1,NOVARS)
20043 FORMAT(//,' **DELTA.4**',10(/,1X,6F20.10))
30120 CONTINUE
C *****
C CHECK TUPLE VALUES AGAINST THE BOUNDS
C        IF ANY
      DO 1100 I=1,NOVARS
```

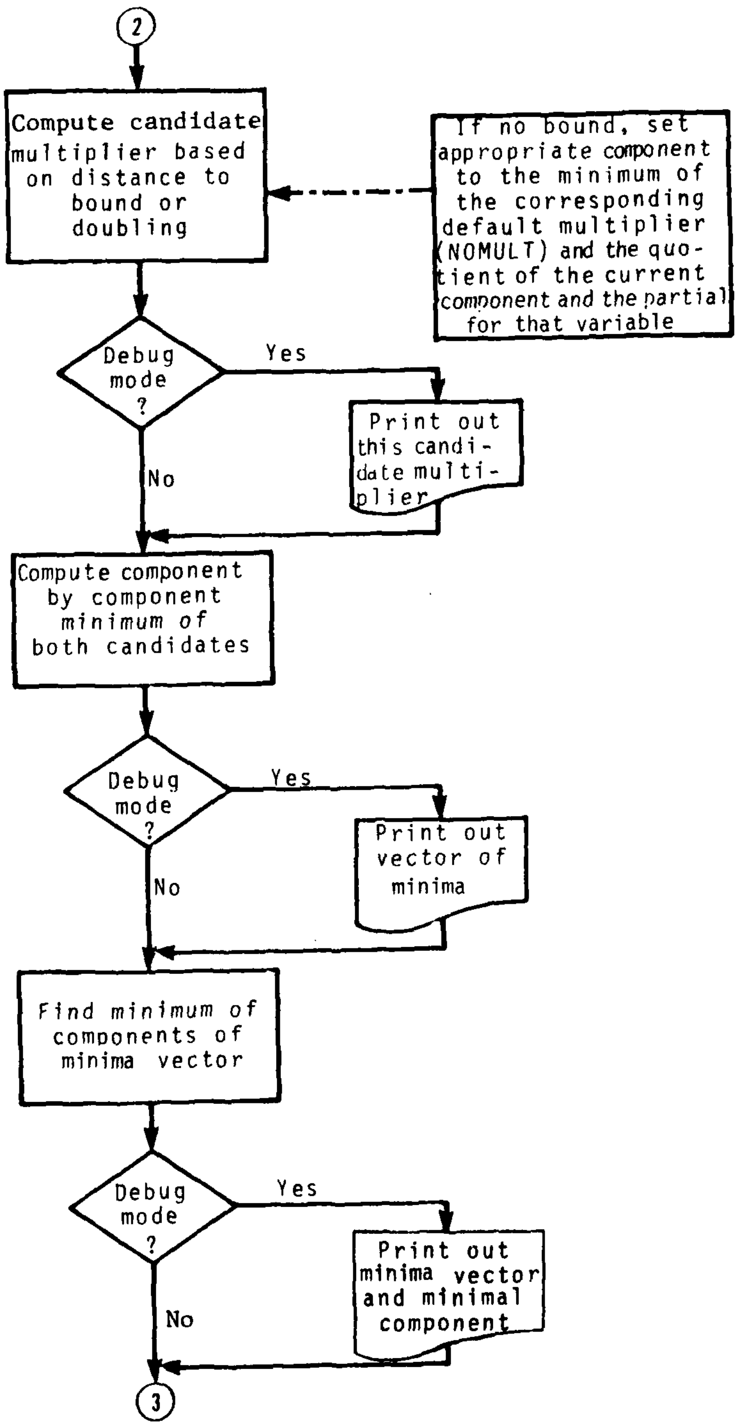
```
      TUPLES(P,I)=TUPLES(P-1,I)+DELTA(P-1,I)
      IF (ABS(HYBDS(I)-NOLIM) .LT. EPSLON) GOTO 1000
      IF (TUPLES(P,I) .GT. HYBDS(I)) TUPLES(P,I)=HYBDS(I)
1000  IF (ABS(LOWBDS(I)-NOLIM) .LT. EPSLON) GOTO 1100
      IF (TUPLES(P,I) .LT. LOWBDS(I)) TUPLES(P,I)=LOWBDS(I)
1100  CONTINUE
C *****
      IF (SWITCH .NE. -1) GOTO 30140
      WRITE(OUTFIL,20047) (TUPLES(P,I),I=1,NOVARS)
20047 FORMAT(//,' **TUPLES**',10(/,1X,6F20.10))
30140 CONTINUE
C *****
      DO 1200 I=1,NOVARS
      INPUT(I)=TUPLES(P,I)
1200  CONTINUE
      CALL PREPR(INPUT,VALUE)
      TUPLES(P,NOVAL1)=VALUE
C   CHECK FOR A SMALL GRADIENT
      SUM=0.
      DO 1300 I=1,NOVARS
      SUM=SUM+GRAD(P-1,I)**2
1300  CONTINUE
      SUM=SUM**(.5)
      Q=P-1
      IF (SUM .LT. FLAT) Q=Q-1
      IF (SUM .LT. FLAT) GOTO 3100
C TO EXIT IF NO SIGNIFICANT CHANGE FROM PREVIOUS VALUES
      DO 3600 I=1,NOVARS
      IF (ABS(TUPLES(P,I)-TUPLES(P-1,I)) .GE. EPSLON) GOTO 3700
3600  CONTINUE
      Q=Q-1
      GOTO 3100
3700  CONTINUE
C END OF POINT COMPUTATION LOOP
3000  CONTINUE
3100  P=Q+1
      DO 1400 I=1,NOVARS
      INPUT(I)=TUPLES(P,I)
1400   CONTINUE
      DO 1500 I=1,NOVARS
      CALL PARTL(NOVARS,I,INPUT,EPSLON,PARTYL,INCR,TUPLES(P,NOVAL1))
C *****
      IF (SWITCH .NE. -1) GOTO 30150
      WRITE(OUTFIL,20030) (INPUT(J),J=1,NOVARS),PARTYL,INCR,
     1   TUPLES(P,NOVAL1)
20030 FORMAT(//,' **LAST INPUT PARTIAL**',10(/,1X,6F20.10))
30150 CONTINUE
```

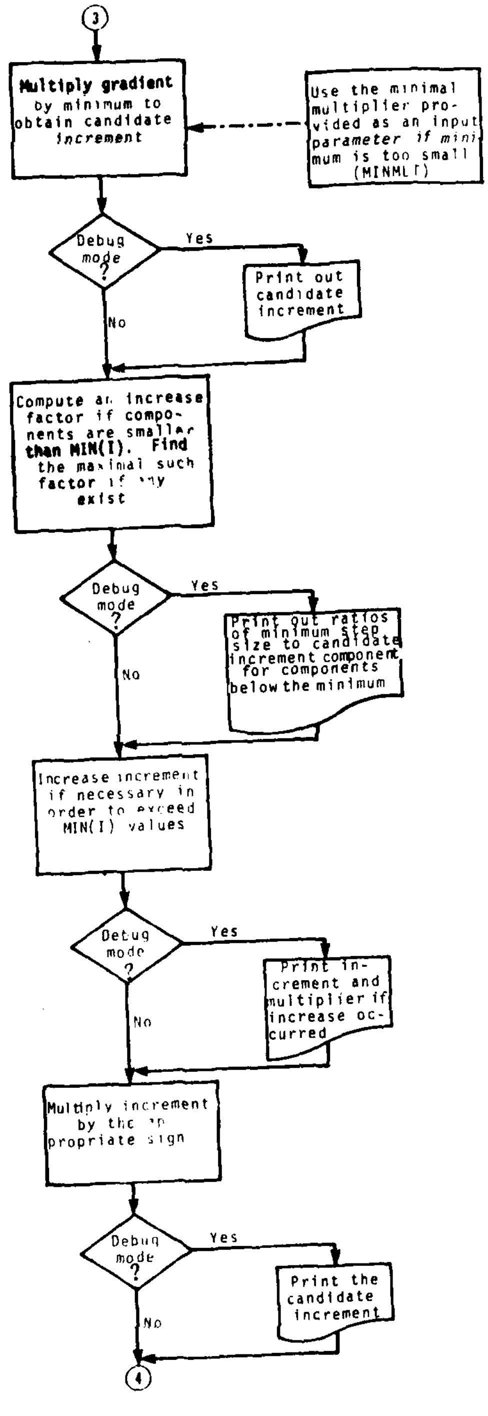
```
C *****
      GRAD(P,I)=PARTYL
1500   CONTINUE
      IF (Q .LT. 1) Q=1
C *****
      IF (SWITCH .NE. -1) GOTO 30130
      WRITE(OUTFIL,20050) SUM
20050 FORMAT(//,' **SUM**',F30.15)
30130 CONTINUE
C *****
C   BEGIN OUTPUT OF COMPUTATIONS
C WRITE FILE FOR VARIABLE VARIATION ROUTINE (VARYVAR)
      WRITE(FILE1,10020) P,NOVARS,EPSLON
10020 FORMAT(I5,I3,F10.7)
      WRITE(FILE1,10030) FORMT2
10030 FORMAT(A80)
      DO 3200 I=1,P
      WRITE(FILE1,FORMT2) (TUPLES(I,J),J=1,NOVARS),TUPLES(I,NOVAL1)
3200  CONTINUE
C   OUTPUT TO PRINTER
      WRITE(OUTFIL,10040)
10040 FORMAT(///,21X,'TUPLES',/)
      DO 3300 I=1,P
      WRITE(OUTFIL,FORMT3) (TUPLES(I,J),J=1,NOVARS),TUPLES(I,NOVAL1)
3300  CONTINUE
      WRITE(OUTFIL,10050)
10050 FORMAT(///,21X,'GRADIENTS',/)
      DO 3400 I=1,P
      WRITE(OUTFIL,FORMT3) (GRAD(I,J),J=1,NOVARS)
3400  CONTINUE
      WRITE(OUTFIL,10060)
10060 FORMAT(///,21X,'INCREMENTS',/)
      DO 3500 I=1,Q
      WRITE(OUTFIL,FORMT3) (DELTA(I,J),J=1,NOVARS)
3500  CONTINUE
      WRITE(FILE2,10070) P,NOVARS,EPSLON
10070 FORMAT(I5,I3,F10.7)
      WRITE(FILE2,10030) FORMT2
      DO 3800 I=1,P
      WRITE(FILE2,FORMT2) (GRAD(I,J),J=1,NOVARS)
3800  CONTINUE
      WRITE(OUTFIL,10090)
10090 FORMAT(///)
      END
```

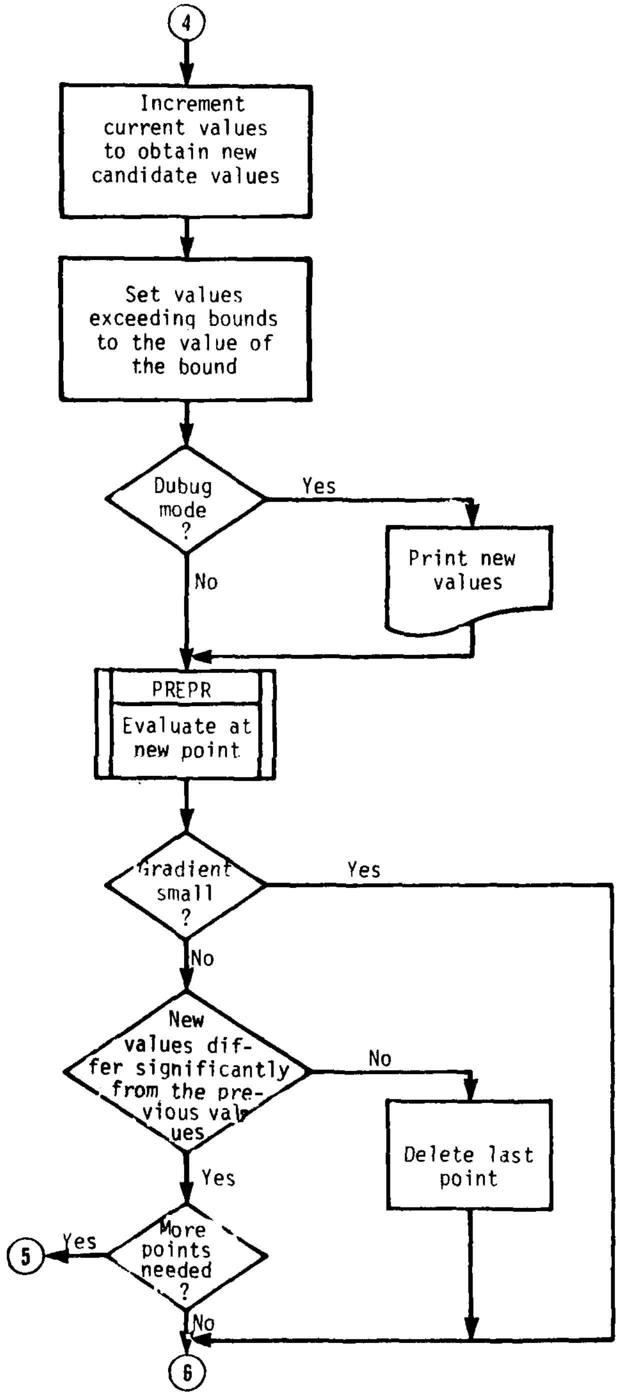
2-8. POINTCOMP ROUTINE FLOWCHART

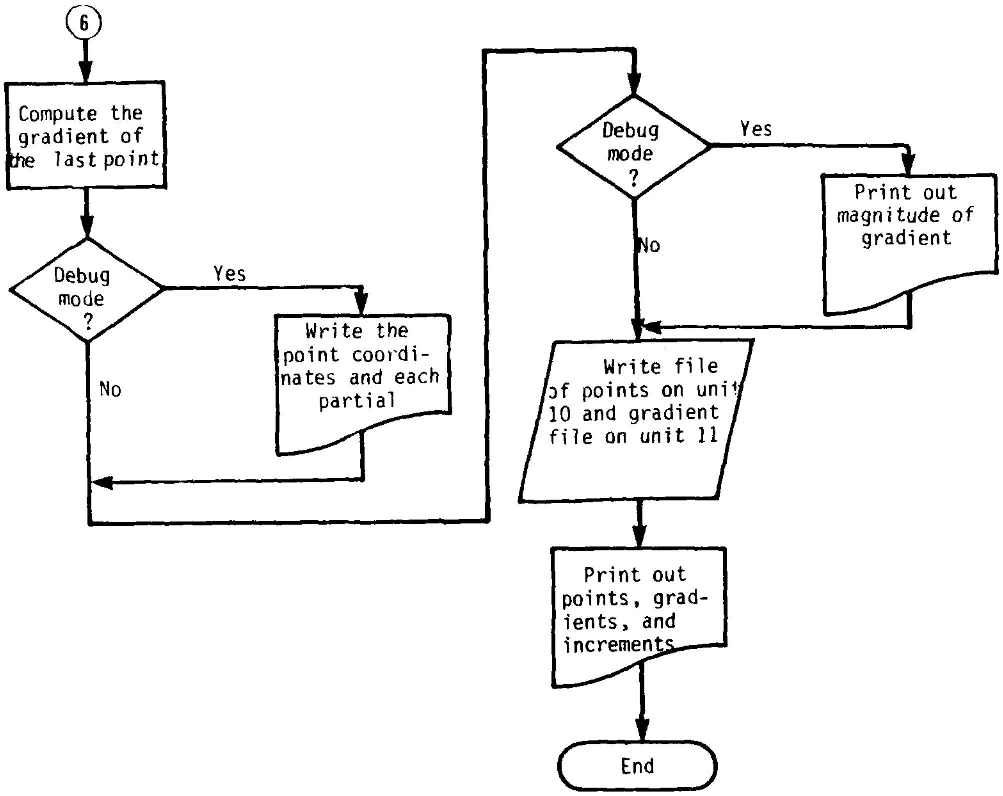



1
PREPR
Call program to be tested to evaluate it on the initial point
5
PARTL
Call routine to compute partial
Debug mode ?
Yes
Print out partial and point computed at
No
Compute the sign of the increments
Debug mode ?
Yes
Print out gradient and sign of each component of increment
No
Compute candidate gradient multiplier based on max step size (MAXCHG)
For variables with gradient component too small, take the default multiplier (NOMULT)
Debug mode ?
Yes
Print out the candidate multiplier
No
2


2
Compute candidate multiplier based on distance to bound or doubling
If no bound, set appropriate component to the minimum of the corresponding default multiplier (NOMULT) and the quotient of the current component and the partial for that variable
Debug mode ?
Yes
Print out this candidate multiplier
No
Compute component by component minimum of both candidates
Debug mode ?
Yes
Print out vector of minima
No
Find minimum of components of minima vector
Debug mode ?
Yes
Print out minima vector and minimal component
No
3


3
Multiply gradient by minimum to obtain candidate increment
Use the minimal multiplier provided as an input parameter if minimum is too small (MINMLT)
Debug mode ?
Yes
Print out candidate increment
No
Compute an increase factor if components are smaller than MIN(I). Find the maximal such factor if any exist
Debug mode ?
Yes
Print out ratios of minimum step size to candidate increment component for components below the minimum
No
Increase increment if necessary in order to exceed MIN(I) values
Debug mode ?
Yes
Print increment and multiplier if increase occurred
No
Multiply increment by the appropriate sign
Debug mode ?
Yes
Print the candidate increment
No
4


4
Increment current values to obtain new candidate values
Set values exceeding bounds to the value of the bound
Dubug mode ?
Yes
Print new values
No
PREPR
Evaluate at new point
Gradient small ?
Yes
No
New values differ significantly from the previous values
No
Delete last point
Yes
More points needed ?
Yes
5
No
6


6
Compute the gradient of the last point
Debug mode ?
Yes
No
Write the point coordinates and each partial
Debug mode ?
Yes
No
Print out magnitude of gradient
Write file of points on unit 10 and gradient file on unit 11
Print out points, gradients, and increments
End

Section II. THE VARVARY1 ROUTINE

2-9. INTRODUCTION. The purpose of this routine is to take each of the points generated by the POINTCOMP routine or chosen by the user and to vary each variable in turn (using the bound and increment) while keeping the other variables fixed. The program to be tested is evaluated at each of these new points, and the new points and their associated program values and gradients are printed out. Difference quotients are computed and printed for each variable as it is varied. The points at which the difference quotient is computed are those defined by the user provided increments and lower bounds. The new points can be used to study the effects of varying just one variable at a time and to provide points for graphs.

2-10. LIMITATIONS

a. When varying a single variable, the only values tested are those of the form: lower bound + integer x increment $\leq$ higher bound. The current routine will not accept a list of values of the form 1, 3, 5, 17, 98.

b. All variables are real.

c. Currently restricted to 100 variables.

d. Currently restricted to a maximum of 500 steps per variable being varied.

e. When this routine is used on output from POINTCOMP or GRID, the limitations of these routines apply as well.

2-11. RUN SETUP

a. Developing an Absolute File in ASCI

```
@MAP,S      , name of absolute element

IN O3PROGTEST.VARVARY1

IN O3PROGTEST.PARTIAL

IN element containing PREPR

IN program to be tested

LIB$*FTN.8.

END
```

b. To Execute

@USE 10, name of file containing input points

@USE 11, name of file into which gradients are placed in standard format

@USE 12, name of file into which new points are placed in standard format

@USE 13, name of scratch file

@USE 14, name of scratch file

@ASG,A name of file containing input points

@ASG,A name of file into which gradients will be placed

@ASG,A name of file into which new points are placed

@ASG,A name of scratch file

@ASG,A name of scratch file

@XQT absolute element

[Input deck]

c. Description of Input Deck

(1) Line 1. (Format to read in each of the following three lines, one at a time.)

(2) Line 2. Lower bounds for the variables; the leftmost bound pertains to the first variable. The line must adhere to the above format.

(3) Line 3. Upper bounds for the variables.

(4) Line 4. Increments for the variables.

(5) Line 5. (The format to print out the point identification number, the new variable values and the output value from the routine being tested, one point at a time.) This format is also used to print out the gradient identification number and the gradient coordinates.

(6) Line 6. (The format to print out one line of difference quotients.)

d. Sample Input Data for VARVARY1

(13F5.2)				Format for reading in the next three lines, one at a time.
5.0	0.5	0.5	0.5	Lower bounds.
18.0	9.0	48.0	1.0	Upper bounds.
3.5	1.5	10.0	0.1	Increments

The leftmost column in rows 1-4 pertains to the first variable, the middle column to the second variable, etc.

e. Sample Input File for VARVARY1--Unit 10

(' POINT NUMBER ',I5,2X,8F10.5) Format for printing out new points and gradients, one to a line.

(1X,13F10.5) Format to print out one line of difference quotients.

INPUT VALUES

```
   9   4  .0010000
(6(7X,F13.5))
  5.00000        .50000        .50000        .50000        .73821
  5.00000        .61902        .75809        .63333        .99916
  5.00119        .73820        .97516        .76667       1.24159
  5.00270        .80924       1.08781        .84667       1.37902
  5.00270        .84061       1.13401        .88222       1.43833
  5.00270        .85178       1.14995        .89492       1.45926
  5.00270        .85513       1.15468        .89873       1.46551
  5.00270        .85513       1.15590        .89873       1.46626
  5.00270        .85513       1.15715        .89975       1.46753
```

This file was created by POINTCOMP on unit 10.

(1) Line 1. The first number is the number of points in the I5 format and the second is the number of variables in an I3 format. The third number is used to determine when a number is essentially zero, in F10.7 format.

(2) Line 2. The format with which the following lines were written, and with which they may be read.

(3) Lines 3-7. The leftmost four columns represent the values of variables one to four, reading from left to right. Each entry in the rightmost column gives the tested program value when run with the variable values printed on the same row. Each row represents an input point. The leftmost four columns are values of the input variable. The last column is the output value of the routine being evaluated.

f. Sample VARVARY1 Run Setup

```
@USE 10,03MAT1.

@ASG,A 03MAT1.

@USE 11,03MAT3.

@ASG,A 03MAT3.

@USE 12,03MAT4.

@ASG,A 03MAT4.

@USE 13,03MAT5.

@ASG,A 03MAT5.

@USE 14,03MAT6.

@ASG,A 03MAT6.

@XQT 03PROGTEST.VARVARY1
```

```
 Input cards:
          (13F5.2)
  5.0  0.5  0.5  0.5
   18.0  9.0 48.0  1.0
  3.5  1.5 10.0  0.1
(' POINT NUMBER ',I5,2X,8F10.5)
(1X,13F10.5)
```

2-12. OUTPUT DESCRIPTION AND SAMPLE OUTPUT

a. Printer

(1) Sample VARVARY1 Output--Points

--------------------VARYING VARIABLE NUMBER 1--------------------

POINT NUMBER	1	5.00000	.50000	.50000	.50000	.73821
POINT NUMBER	2	8.50000	.50000	.50000	.50000	.73934
POINT NUMBER	3	12.00000	.50000	.50000	.50000	.73985
POINT NUMBER	4	15.50000	.50000	.50000	.50000	.74014
POINT NUMBER	5	19.00000	.50000	.50000	.50000	.74033

+++++++++++++++++++DIFFERENCE QUOTIENT MATRIX++++++++++++++++++

.00000				
.00032	.00000			
.00023	.00015	.00000		
.00018	.00011	.00008	.00000	
.00015	.00009	.00007	.00005	.00000

--------------------VARYING VARIABLE NUMBER 2--------------------

POINT NUMBER	6	5.00000	.50000	.50000	.50000	.73821
POINT NUMBER	7	5.00000	2.00000	.50000	.50000	1.10039
POINT NUMBER	8	5.00000	3.50000	.50000	.50000	1.32998
POINT NUMBER	9	5.00000	5.00000	.50000	.50000	1.48843
POINT NUMBER	10	5.00000	6.50000	.50000	.50000	1.60434
POINT NUMBER	11	5.00000	8.00000	.50000	.50000	1.69280
POINT NUMBER	12	5.00000	9.50000	.50000	.50000	1.76252

+++++++++++++++++++DIFFERENCE QUOTIENT MATRIX++++++++++++++++++

.00000					
.24145	.00000				
.19726	.15306	.00000			
.16672	.12935	.10564	.00000		
.14436	.11199	.09145	.07727	.00000	
.12728	.09874	.08063	.06812	.05897	.00000

(2) Sample VARVARY1 Output--Gradients

THE VARIABLE VARIED IS NUMBER 1

POINT NUMBER	1	.00048	.30919	.67048	.34638
POINT NUMBER	2	.00019	.32471	.68200	.34773
POINT NUMBER	3	.00010	.33081	.68706	.34836
POINT NUMBER	4	.00006	.33495	.68991	.34871
POINT NUMBER	5	.00004	.33762	.69173	.34894

THE VARIABLE VARIED IS NUMBER 2

POINT NUMBER	6	.00048	.30919	.67048	.34638
POINT NUMBER	7	.02158	.18565	.60644	1.08123
POINT NUMBER	8	.05104	.12313	.56450	1.55265
POINT NUMBER	9	.07778	.08812	.53491	1.88027
POINT NUMBER	10	.10199	.06543	.51291	2.12195
POINT NUMBER	11	.12182	.05001	.49591	2.30725
POINT NUMBER	12	.13869	.04014	.48239	2.45384

(3) Description of Points Output. Each row describes a point. The leftmost four columns are values of variables one through four, reading from left to right. The rightmost entry on each row is the output value when the inputs are those in columns one through four.

(4) Description of Gradients Output. Each row is the gradient of the routine being tested at the point whose number is listed on the left. Again, the values of variables one to four are listed from left to right. The point numbers link the printout of the points and variable values to the gradient printout.

(5) Sample VARVARY1 Output--Difference Quotients

GIVEN SAMPLE OUTPUT

POINT NUMBER 51	5.00000	.73820	.97516	.76667	1.24158
POINT NUMBER 52	8.50000	.73820	.97516	.76667	1.26175
POINT NUMBER 53	12.00000	.73820	.97516	.76667	1.27087
POINT NUMBER 54	15.50000	.73820	.97516	.76667	1.27607
POINT NUMBER 55	19.00000	.73820	.97516	.76667	1.27942

We define the following labels for some of the variable values and outputs:

Values	Outputs
$X_1 = 5.00000$	$Y_1 = 1.24158$
$X_2 = 8.50000$	$Y_2 = 1.26175$
$X_3 = 12.00000$	$Y_3 = 1.27087$
$X_4 = 15.50000$	$Y_4 = 1.27607$
$X_5 = 19.00000$	$Y_5 = 1.27942$

THE ASSOCIATED DIFFERENCE QUOTIENTS ARE

.00000				
.00576	.00000			
.00418	.00261	.00000		
.00328	.00204	.00148	.00000	
.00270	.00168	.00122	.00096	.00000

If we denote the varying coordinate of point numbers 51-55 by x_1 to x_5, and denote the corresponding values (the rightmost column) by y_1 to y_5, the second row in the matrix has the nonzero entry:

$$\frac{y_2 - y_1}{x_2 - x_1} = .00576.$$

Note that the difference in y is in the numerator and the difference in x is in the denominator. The third row has the nonzero entries:

$$\frac{y_3 - y_1}{x_3 - x_1} = .00418$$

$$\frac{y_3 - y_2}{x_3 - x_2} = .00261$$

Note that the third row has the first term of numerators and denominators indexed by 3 and the second terms are indexed by 1 and 2 (i.e., 3-1). The fourth row has the nonzero entries:

$$\frac{y_4 - y_1}{x_4 - x_1} = .00328$$

$$\frac{y_4 - y_2}{x_4 - x_2} = .00204$$

$$\frac{y_4 - y_3}{x_4 - x_3} = .00148$$

Note that the fourth row has the first term in each numerator and denominator indexed by 4 and the second terms are indexed by 1, 2, and 3 (i.e., 4-1). In general, the nth row is comprised of the ordered set:

$$\left(\frac{y_n - y_j}{x_n - x_j} \mid j=1,\ldots,n-1\right)$$

To show the geometrical significance of these computations given the following points:

POINT NUMBER 51	5.00000	.73820	.97516	.76667	1.24158
POINT NUMBER 52	8.50000	.73820	.97516	.76667	1.26175
POINT NUMBER 53	12.00000	.73820	.97516	.76667	1.27087
POINT NUMBER 54	12.50000	.73820	.97516	.76667	1.27607
POINT NUMBER 55	19.00000	.73820	.97516	.76667	1.27942

and the difference quotients:

.00000						
.00576		.00000				
.00418	A	.00261	.00000			
.00328		.00204	.00148		.00000	
.00270		.00168	.00122	B	.00096	.00000

Table 2-1. Table of Difference Quotients

The purpose of these computations will now be explained. The import of the quotients in triangle A is illustrated in Figure 2-1 (drawn not to scale).

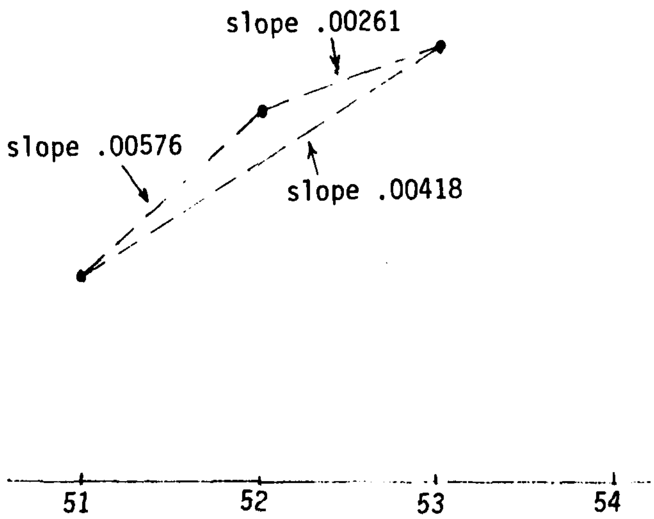


Figure 2-1. Triangle A in Table 2-1

The figures in triangle B show that:

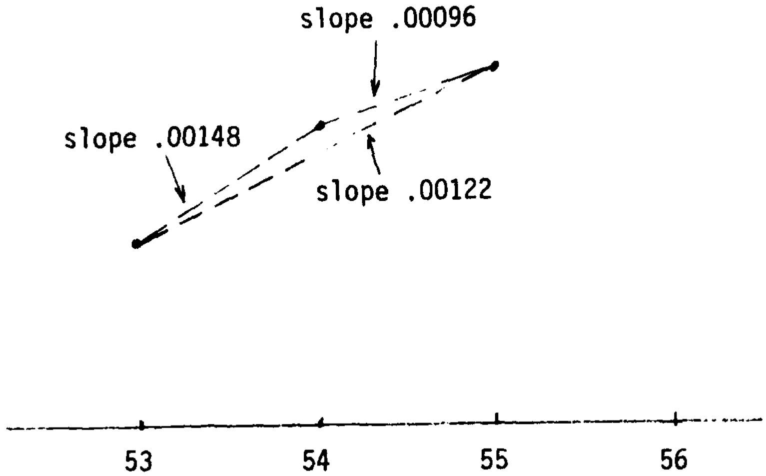


Figure 2-2. Triangle B in Table 2-1

The fourth line gives the following picture (not to scale):

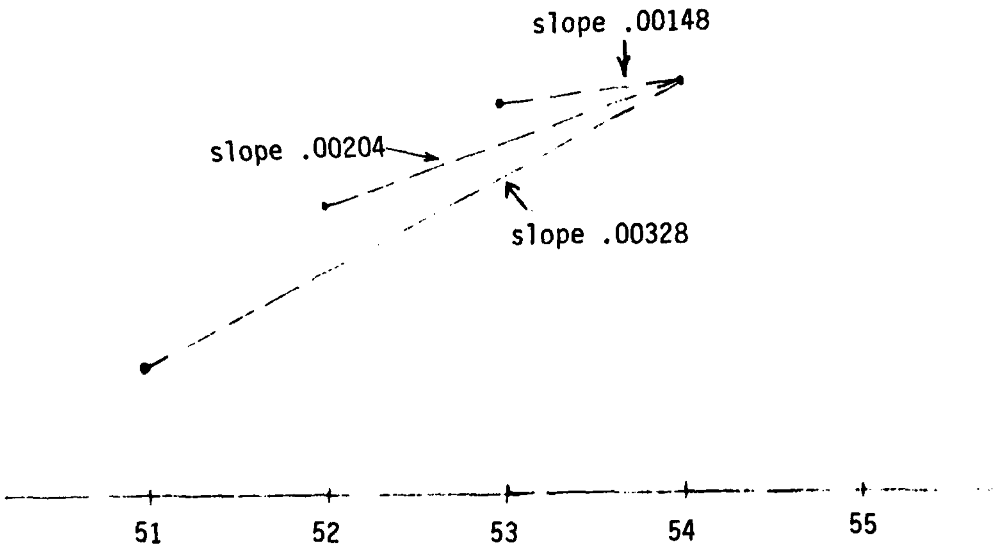


Figure 2-3. Slopes on the Fourth Line of Table 2-1

Note that since this is the fourth row, all lines have the fourth point as terminus, reading from left to right. The second column gives the following picture (not to scale):

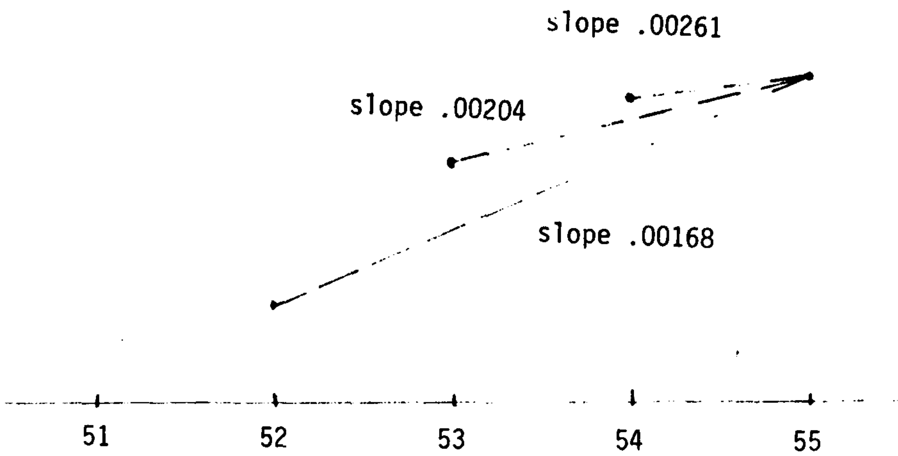


Figure 2-4. Slopes on the Second Column of Table 2-1

Note that since this is the second column, all lines start at point number 52, reading from left to right.

b. <u>Sample VARVARY1 Output Files and Descriptions</u>

(1) <u>Standard Format File on Unit 12--Points</u>

```
  225  4  .0010000
(6(7X,F13.5))
      5.00000       .50000       .50000       .50000       .73821
      8.50000       .50000       .50000       .50000       .73934
     12.00000       .50000       .50000       .50000       .73985
     15.50000       .50000       .50000       .50000       .74014
     19.00000       .50000       .50000       .50000       .74033
```

The first row indicates that there are 225 points in the file, that there are four variables (the rightmost column gives the output values), and that numbers smaller than .001 are considered to be zero. The second row gives the format in which the file was written, which may be used for reading in the file. The 225 points and values commence at line 3 and comprise the remainder of the file. (Only five points are illustrated.)

(2) <u>Standard Format File on Unit 11--Gradients</u>

```
  225  4  .0010000
(6(7X,F13.5))
      .00048          .30919          .67048          .34638
      .00019          .32471          .68200          .34773
      .00010          .33081          .68706          .34836
      .00006          .33495          .68991          .34871
      .00004          .33762          .69173          .34894
```

The first two rows of this file are described as above. The gradients comprise the following rows.

(3) <u>Scratch Files</u>

(a) <u>Unit 13</u>

```
-19.00100    4.00100    .00048    .30919    .67048    .34638
 20.00100    4.00100    .00055    .37265    .66454    .34684
 21.00100    4.00100    .00061    .43536    .65859    .34731
 22.00100    4.00100    .00067    .49823    .65264    .34778
 23.00100    4.00100    .00072    .56128    .64668    .34825
 24.00100    4.00100    .00076    .62451    .64071    .34872
 25.00100    4.00100    .00079    .68790    .63473    .34920
-26.00100    1.00100    .00359    .35940    .65458    .40207
```

The first column is a point count. A negative sign indicates a new variable is being varied. The second column indicates which variable is being varied. Columns 3 through 6 are the gradient values for variables 1 to 4, respectively.

(b) Unit 14

5.00000	.50000	.50000	.50000	.73821
8.50000	.50000	.50000	.50000	.73934
12.00000	.50000	.50000	.50000	.73985
15.50000	.50000	.50000	.50000	.74014
19.00000	.50000	.50000	.50000	.74033
5.00000	.50000	.50000	.50000	.73821
5.00000	2.00000	.50000	.50000	1.10039

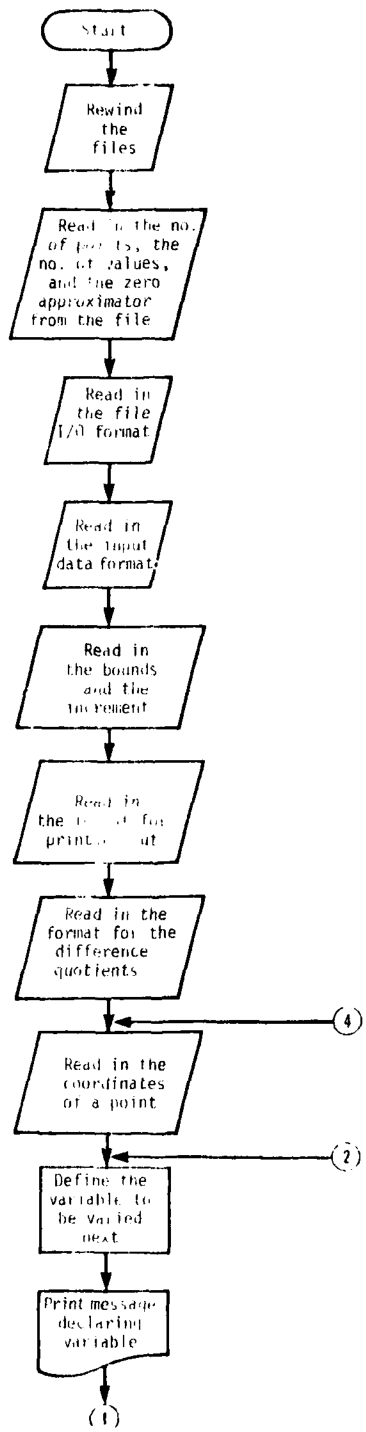
The first four columns indicate variable values. The fifth column gives the output values, each one corresponding to the input values listed on the same row.

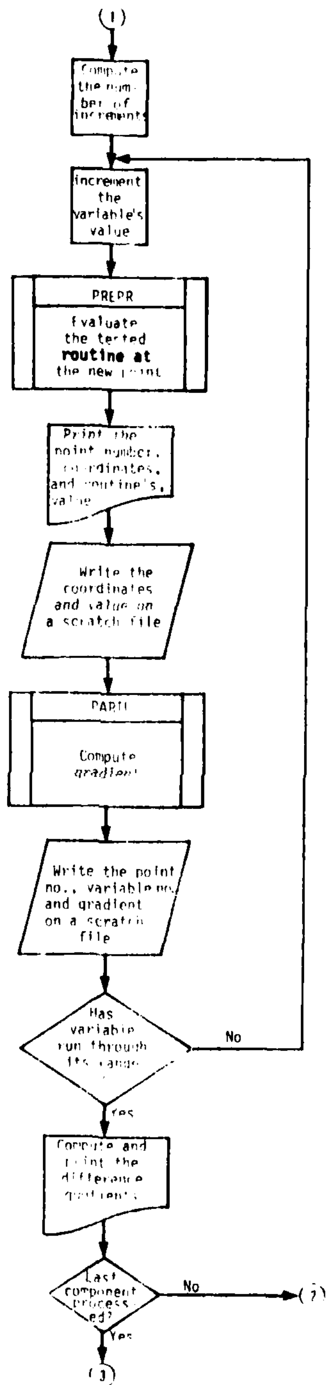
2-13. VARVARY1 ROUTINE LISTING

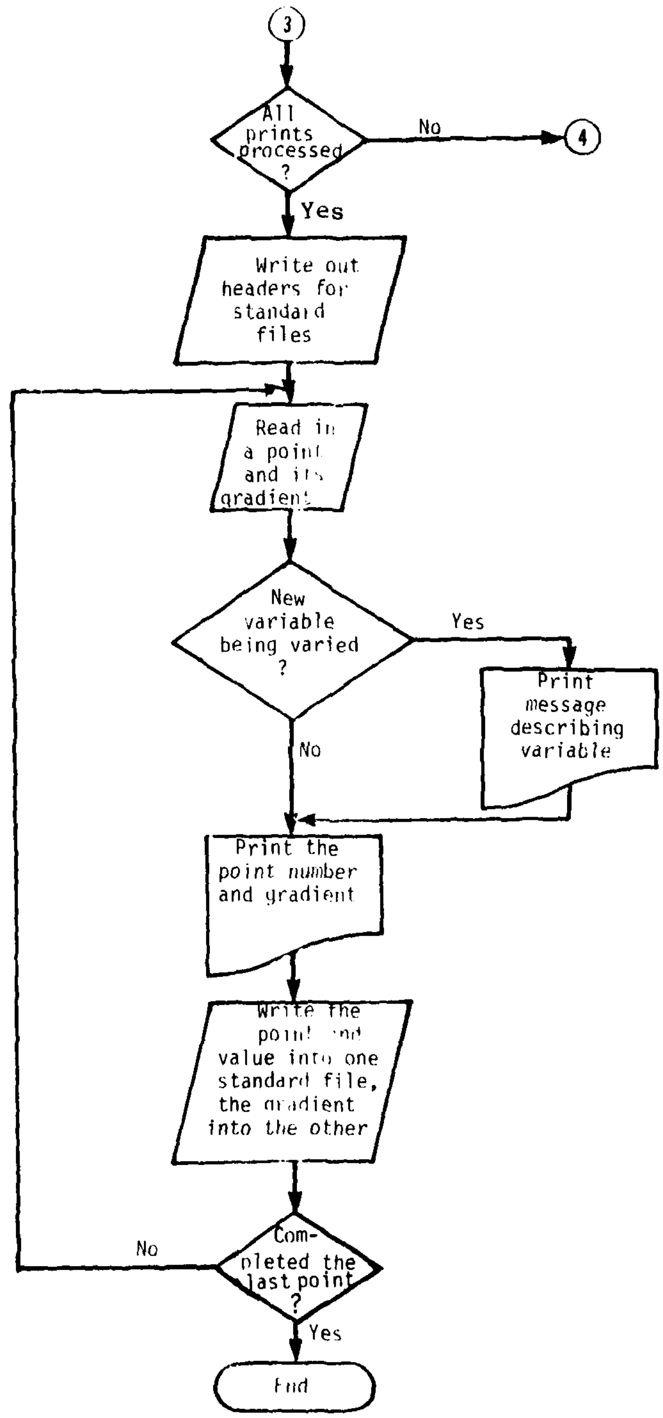
```
      PARAMETER FILE1=10,FILE4=13,NOVALS=100,INFILE=5,OUTFIL=6
      PARAMETER POINTS=500,FILE2=11,FILE3=12,FILE5=14
C
      DIMENSION TUPLES(NOVALS),LOWBDS(NOVALS),HYBDS(NOVALS),INCR(NOVALS)
      DIMENSION VALUES(NOVALS),GRAD(NOVALS),VALU(POINTS),LINE(POINTS)
C
      REAL TUPLES,LOWBDS,HYBDS,INCR,VALUES,VALUE,VALU,LINE,INCR1
      REAL PARTYL,FE,FG,EPSLON
C
      INTEGER NOVARS,NOPTS,I,J,K,L,P,Q
C
      CHARACTER*80 FORMT1,FORMT2,FORMT3,FORMT4
C
      REWIND FILE1
      REWIND FILE2
      REWIND FILE3
      REWIND FILE4
      REWIND FILE5
      Q=0
      READ(FILE1,10000) NOPTS,NOVARS,EPSLON
10000 FORMAT(I5,I3,F10.7)
      READ(FILE1,10010) FORMT1
10010 FORMAT(A80)
      READ(INFILE,10010) FORMT2
      READ(INFILE,FORMT2) (LOWBDS(I),I=1,NOVARS)
      READ(INFILE,FORMT2) (HYBDS(I),I=1,NOVARS)
      READ(INFILE,FORMT2) (INCR(I),I=1,NOVARS)
      READ(INFILE,10010) FORMT3
      READ(INFILE,10010) FORMT4
      WRITE(OUTFIL,10030)
10030 FORMAT(/////)
      DO 400 P=1,NOPTS
      READ(FILE1,FORMT1) (TUPLES(I),I=1,NOVARS)
      DO 300 I=1,NOVARS
      WRITE(OUTFIL,10070) I
10070 FORMAT(1X,10(1H-),'VARYING VARIABLE NUMBER ',I3,10(1H-),//)
      DO 500 J=1,NOVARS
      VALUES(J)=TUPLES(J)
500   CONTINUE
100   K=INT((HYBDS(I)-LOWBDS(I))/INCR(I))+1
      DO 200 J=0,K
      VALUES(I)=LOWBDS(I)+J*INCR(I)
      CALL PREPR(VALUES,VALUE)
      Q=Q+1
      WRITE(OUTFIL,FORMT3) Q,(VALUES(L),L=1,NOVARS),VALUE
      WRITE(FILE5,FORMT1) (VALUES(L),L=1,NOVARS),VALUE
      VALU(J+1)=VALUE
```

```
      FQ=Q+EPSLON
      FI=I+EPSLON
      DO 150 L=1,NOVARS
      CALL PARTL(NOVARS,L,VALUES,EPSLON,PARTYL,INCR1,VALUE)
      GRAD(L)=PARTYL
150   CONTINUE
      IF (J .EQ. 0) FQ=-FQ
      WRITE(FILE4,FORMT1) FQ,FI,(GRAD(L),L=1,NOVARS)
200   CONTINUE
      WRITE(OUTFIL,10050)
10050 FORMAT(/,1X,20(1H+),'DIFFERENCE QUOTIENT MATRIX',20(1H+),/)
      DO 700 J=0,K
      DO 600 L=0,J
      LINE(L+1)=(VALU(J+1)-VALU(L+1))/(MAX0(J-L,1)*INCR(I))
600   CONTINUE
      WRITE(OUTFIL,FORMT4) (LINE(L+1),L=0,J)
700   CONTINUE
      WRITE(OUTFIL,10060)
10060 FORMAT(/)
300   CONTINUE
      WRITE(OUTFIL,10030) P
10030 FORMAT(/,1X,30(1H/),'END OF COMPUTATIONS FOR POINT NUMBER ',
     1  I5,3X,30(1H/),///)
400   CONTINUE
      WRITE(FILE2,10000) Q,NOVARS,EPSLON
      WRITE(FILE2,10010) FORMT1
      WRITE(FILE3,10000) Q,NOVARS,EPSLON
      WRITE(FILE3,10010) FORMT1
      REWIND FILE4
      REWIND FILE5
      WRITE(OUTFIL,10090)
10090 FORMAT(//,1X,30(1H ),'THE CORRESPONDING GRADIENTS',/)
      DO 800 P=1,Q
      READ(FILE4,FORMT1) FQ,FI,(GRAD(L),L=1,NOVARS)
      READ(FILE5,FORMT1) (VALUES(L),L=1,NOVARS),VALUE
      IF (FQ .GT. 0.) GOTO 750
      FQ=-FQ
      K=INT(FI)
      WRITE(OUTFIL,10100) K
10100 FORMAT(/,10(1H ),' THE VARIABLE VARIED IS NUMBER ',I5,/)
750   J=INT(FQ)
      WRITE(OUTFIL,FORMT3) J,(GRAD(L),L=1,NOVARS)
      WRITE(FILE3,FORMT1) (VALUES(L),L=1,NOVARS),VALUE
      WRITE(FILE2,FORMT1) (GRAD(L),L=1,NOVARS)
800   CONTINUE
      WRITE(OUTFIL,10040)
10040 FORMAT(///)
      END
```

2-14. VARVARY1 ROUTINE FLOWCHART




1
Compute the number of increments
Increment the variable's value
PREPR
Evaluate the tested routine at the new point
Print the point number, coordinates, and routine's value
Write the coordinates and value on a scratch file
PARTL
Compute gradient
Write the point no., variable no., and gradient on a scratch file
Has variable run through its range
No
Yes
Compute and print the difference gradients
Last component processed?
No
2
Yes
3


3
All prints processed ?
No
4
Yes
Write out headers for standard files
Read in a point and its gradient
New variable being varied ?
Yes
Print message describing variable
No
Print the point number and gradient
Write the point and value into one standard file, the gradient into the other
Com-pleted the last point ?
No
Yes
End

CHAPTER 3

THE ANALYZ SUBSYSTEM

3-1. INTRODUCTION. This routine computes variable sensitivity statistics from gradients in a standard format file on unit 11. Statistics are initially computed on each gradient. The gradients are summed, component by component, and the same statistics are computed for the sum vector.

3-2. DISCUSSION OF STATISTICS

a. The first statistic is the ratio of each component of the gradient to the component with the smallest absolute value.

b. The second statistic is the change in the component required to achieve a unit change in the output for each component of the gradient. For the summed vector, a change in the output equal to the number of gradients rather than a unit change is utilized.

c. The third statistic is the ratio of each component to the component with the largest absolute value; except that for the largest component, its ratio with the next largest component is taken.

3-3. LIMITATIONS

a. The gradients input to ANALYZ must all be real.

b. No more than 500 gradients can be analyzed in a single run by the currently compiled version of ANALYZ.

c. No more than 20 variables may be analyzed by the current version of ANALYZ.

3-4. RUN SETUPS

a. <u>To Execute</u>

@USE 11, name of file containing gradients in standard format.

@ASG,A filename.

@XQT O3PROGTEST.ANALYZ

[Input deck]

b. Description of Input Deck

(1) Line 1. Field: A1

Format: I3

A1 is +1 if no debugging information is desired, -1 if debugging is desired.

(2) Line 2. (Format for printing the statistics.)

(3) Line 3. (Format for debug printouts.)

c. Sample Input Data

+1	Indicates no debugging output requested.
(6(7X,F13.5),/,100(21X,5(F13.5,7X),/))	Format for statistical output
(6(7X,F13.5),/,100(21X,5(F13.5,7X),/))	Debugging data output format

d. Input File Description on Unit 11

Produced by POINTCOMP on Unit 11

```
    9   4  .0010000
(6(7X,F13.5))
        .00048        .30919        .67048        .34638
        .00359        .35940        .65458        .40207
        .00852        .40086        .63565        .45144
        .01236        .42295        .62294        .47942
        .01428        .43221        .61698        .49139
        .01500        .43545        .61481        .49560
        .01522        .43641        .61415        .49686
        .01523        .43631        .61415        .49674
        .01527        .43670        .61405        .49663
```

(1) Line 1

(a) Format: I5, I3, F10.7.

(b) Meaning: Nine lines, four variables, zero approximator .001.

(2) Line 2. Format for reading in the following lines of data, one at a time.

(3) Other Lines. Each row is a gradient, variable 1 values are in the leftmost column, variable 2 values are in the next column, etc.

e. Sample Run Setup

```
@USE 11,03MAT2.

@ASG,A 03MAT2.

@XQT 03PROGTEST.ANALYZ
 +1

(6(7X,F13.5),/,100(21X,5(F13.5,7X),/))

(6(7X,F13.5),/,100(21X,5(F13.5,7X),/))
```

3-5. OUTPUT DESCRIPTION AND SAMPLE OUTPUT

a. Sample Output--Gradient by Gradient

STATISTICS ON EACH GRADIENT

************************EQUIVALENT CHANGES***********************

.00000	3.23426	1.48147	2.88700
278.55153	2.78242	1.52770	2.48713
117.37089	2.49464	1.57319	2.21513
80.90615	2.36435	1.60529	2.08585
70.02801	2.31369	1.62080	2.03504
66.66667	2.29647	1.62652	2.01776
65.70302	2.29142	1.62827	2.01264
65.65988	2.29195	1.62827	2.01313
65.48788	2.28990	1.62853	2.01357

****************************COMPARISONS****************************

.00000	.46155	1.93568	.51661
.00548	.54905	1.62802	.61424
.01340	.63063	1.40805	.71020
.01984	.67896	1.29936	.76961
.02314	.70053	1.25558	.79644
.02440	.70827	1.24054	.80610
.02478	.71059	1.23606	.80902
.02480	.71043	1.23636	.80883
.02487	.71118	1.23643	.80878

*************************RELATIVE RATIOS*************************

.00000	1.00000	2.16850	1.12028
1.00000	100.11142	182.33426	111.99721
1.00000	47.04930	74.60681	52.98591
1.00000	34.21926	50.39968	38.78803
1.00000	30.26681	43.20588	34.41106
1.00000	29.03000	40.99733	33.04000
1.00000	28.67346	40.35151	32.64520
1.00000	28.64806	40.32502	32.61589
1.00000	28.59856	40.21284	32.52325

b. Description. The preceding statistics are described in paragraph 3-2. For each statistic, each row corresponds to a gradient.

c. Sample Output--Sum of Gradients

SUMMED GRADIENT COMPUTATIONS

*************************SUMMED GRADIENTS*************************

.09947	3.66948	5.65779	4.15653

********************EQUIVALENT SUMMED CHANGES********************

90.47954	2.45266	1.59073	2.16427

************************SUMMED COMPARISONS************************

.01758	.64857	1.36118	.73466

**********************RELATIVE SUMMED RATIOS**********************

1.00000	36.89032	56.87936	41.78677

d. Description. The first row is the sum of the gradients. The last three rows are the same statistics as before, in the same order, but applied to the sum vector only.

3-6. ANALYZ ROUTINE LISTING

```
        PARAMETER FILE2=11,INFILE=5,OUTFIL=6
        PARAMETER NOVALS=20,NOGRAD=500
C
        DIMENSION GRAD(NOGRAD,NOVALS),RMINS(NOGRAD),CSUMS(NOVALS)
        DIMENSION RMAXS(NOGRAD,2),CSMAXS(2),PRINT(NOVALS)
C
        REAL GRAD,RMINS,CSUMS,RMAXS,CSMAXS,PRINT,EPSLON
C
        INTEGER C,I,V,NOVARS,NOGRD,SWITCH
C
        CHARACTER*80 FMTRD,FMTWRT,FMTDBG
C
C READ THE INPUTS
C
        READ(INFILE,100) SWITCH
  100   FORMAT(I3)
        READ(INFILE,200) FMTWRT
  200   FORMAT(A80)
C
C READ IN THE GRADIENT VALUES
C
        REWIND FILE2
        READ(FILE2,300) NOGRD,NOVARS,EPSLON
  300   FORMAT(I5,I3,F10.7)
        READ(FILE2,200) FMTRD
        DO 400 G=1,NOGRD
        READ(FILE2,FMTRD) (GRAD(G,V),V=1,NOVARS)
  400   CONTINUE
C
C READ IN DEBUG MODE FORMAT IF IN DEBUG MODE
C
        IF (SWITCH .EQ. -1) READ(INFILE,200) FMTDBG
C **********
        IF (SWITCH .NE. -1) GOTO 20200
        WRITE(OUTFIL,20000) NOGRD,NOVARS
  20000 FORMAT(//,' **INPUT**',[illegible])
        DO 20100 C=1,NOGRD
        WRITE(OUTFIL,FMTDBG) (GRAD(G,V),V=1,NOVARS)
  20100 CONTINUE
  20200 CONTINUE
C **********
C
C COMPUTE ROW MINIMA
C
        DO 750 G=1,NOGRD
        DO 500 V=1,NOVARS
        RMINS(G)=0.0
        IF (ABS(GRAD(G,V)) .LT. EPSLON) GOTO 500
        RMINS(G)=ABS(GRAD(G,V))
```

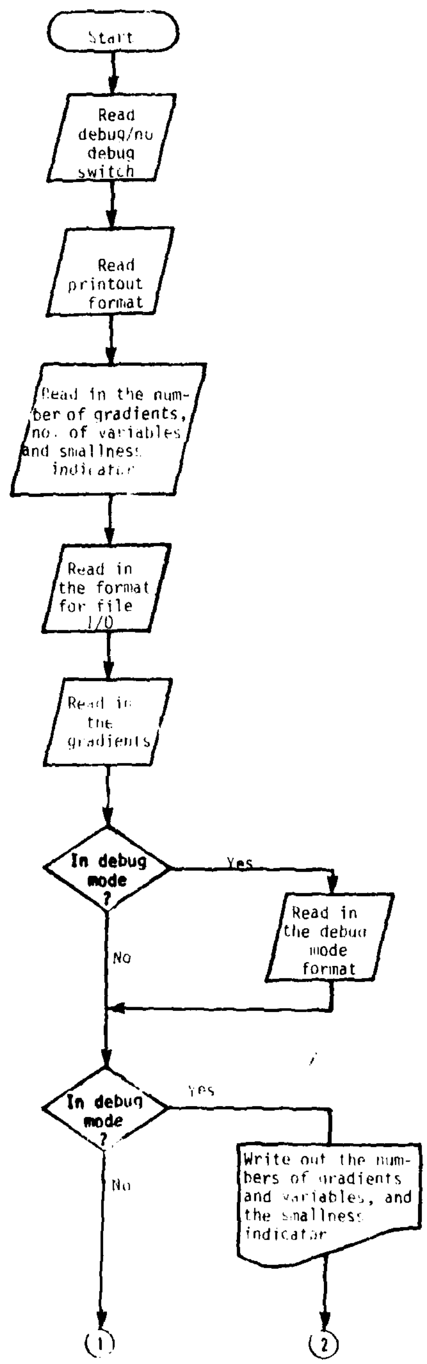
```
      GOTO 600
500   CONTINUE
600   DO 700 V=1,NOVARS
      IF (ABS(GRAD(G,V)) .LT. EPSLON) GRAD(G,V)=0.0
      IF (ABS(GRAD(G,V)) .LT. EPSLON) GOTO 700
      IF (ABS(GRAD(G,V)) .LT. RMINS(G)) RMINS(G)=ABS(GRAD(G,V))
700   CONTINUE
750   CONTINUE
C **********
      IF (SWITCH .NE. -1) GOTO 20400
      WRITE(OUTFIL,20300)
20300 FORMAT(//,' **ROW MINS**',/)
      WRITE(OUTFIL,FMTDBG) (RMINS(G),G=1,NOGRD)
20400 CONTINUE
C **********
C
C COMPUTE THE COLUMN SUMS
C
      DO 900 V=1,NOVARS
      CSUMS(V)=0.
      DO 800 G=1,NOGRD
      CSUMS(V)=CSUMS(V)+GRAD(G,V)
800   CONTINUE
      IF (ABS(CSUMS(V)) .LT. (NOGRD*EPSLON)) CSUMS(V)=0.0
900   CONTINUE
      CSMIN=ABS(CSUMS(1))
      DO 1000 V=2,NOVARS
      IF (ABS(CSUMS(V)) .LT. (NOGRD*EPSLON)) GOTO 1000
      IF (ABS(CSUMS(V)) .LT. CSMIN) CSMIN=ABS(CSUMS(V))
1000  CONTINUE
C
C TO COMPUTE THE TWO DISTINCT LARGEST PARTIALS(IN ABSOLUTE VALUE)
C       IN EACH GRADIENT
C
      IF (NOVARS .LT. 2) GOTO 1600
      DO 1350 G=1,NOGRD
      RMAXS(G,1)=ABS(GRAD(G,1))
      DO 1100 V=2,NOVARS
      IF (ABS(GRAD(G,V)) .GT. RMAXS(G,1)) RMAXS(G,1)=ABS(GRAD(G,V))
1100  CONTINUE
      DO 1150 V=1,NOVARS
      IF (ABS(GRAD(G,V)) .LT. EPSLON) GOTO 1150
      IF (ABS(GRAD(G,V)) .GE. RMAXS(G,1)) GOTO 1150
      RMAXS(G,2)=ABS(GRAD(G,V))
      GOTO 1200
1150  CONTINUE
      RMAXS(G,2)=RMAXS(G,1)
      GOTO 1350
```

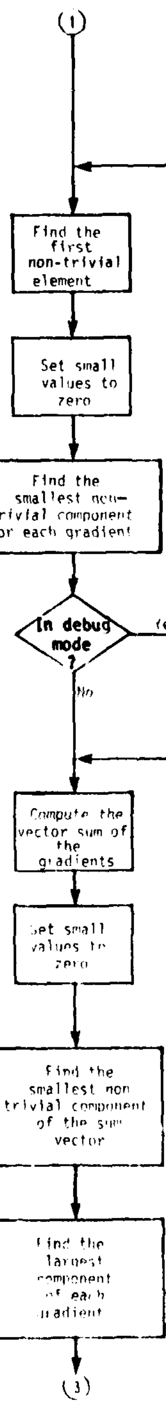
```
1200  DO 1300 V=1,NOVARS
      IF ((ABS(GRAD(G,V)) .GT.RMAXS(G,2)) .AND. (ABS(GRAD(G,V)) .LT.
     1  RMAXS(G,1))) RMAXS(G,2)=ABS(GRAD(G,V))
1300  CONTINUE
1350  CONTINUE
C **********
      IF (SWITCH .NE. -1) GOTO 21000
      WRITE(OUTFIL,20700)
20700 FORMAT(/,' **ROW MAXIMA**',/)
      DO 20900 G=1,NOGRD
      WRITE(OUTFIL,FMTDBG) (GRAD(G,V),V=1,NOVARS),RMAXS(G,1),RMAXS(G,2)
20900 CONTINUE
21000 CONTINUE
C **********
C
C TO COMPUTE THE TWO LARGEST COMPONENTS OF CSUMS IN ABSOLUTE VALUE
C
      CSMAXS(1)=ABS(CSUMS(1))
      DO 1400 V=2,NOVARS
      IF (ABS(CSUMS(V)) .GT. CSMAXS(1)) CSMAXS(1)=ABS(CSUMS(V))
1400  CONTINUE
      DO 1425 V=1,NOVARS
      IF (ABS(CSUMS(V)) .LT. (NOGRD*EPSLON)) GOTO 1425
      IF (ABS(CSUMS(V)) .GE. CSMAXS(1)) GOTO 1425
      CSMAXS(2)=ABS(CSUMS(V))
      GOTO 1450
1425  CONTINUE
      CSMAXS(2)=CSMAXS(1)
      GOTO 1550
1450  DO 1500 V=1,NOVARS
      IF ((ABS(CSUMS(V)) .GT. CSMAXS(2)) .AND. (ABS(CSUMS(V)) .LT.
     1  CSMAXS(1))) CSMAXS(2)=ABS(CSUMS(V))
1500  CONTINUE
1550  CONTINUE
C **********
      IF (SWITCH .NE. -1) GOTO 21300
      WRITE(OUTFIL,21100)
21100 FORMAT(/,' **CSUMS MAXIMA**',/)
      WRITE(OUTFIL,FMTDBG) (CSUMS(V),V=1,NOVARS),CSMAXS(1),CSMAXS(2)
21300 CONTINUE
C **********
1600  CONTINUE
C
C TO COMPUTE THE RELATIVE RATIOS
C
      WRITE(OUTFIL,10000)
10000 FORMAT(//,1X,30(1H*),'RELATIVE RATIOS',30(1H*),/)
      DO 1800 G=1,NOGRD
```

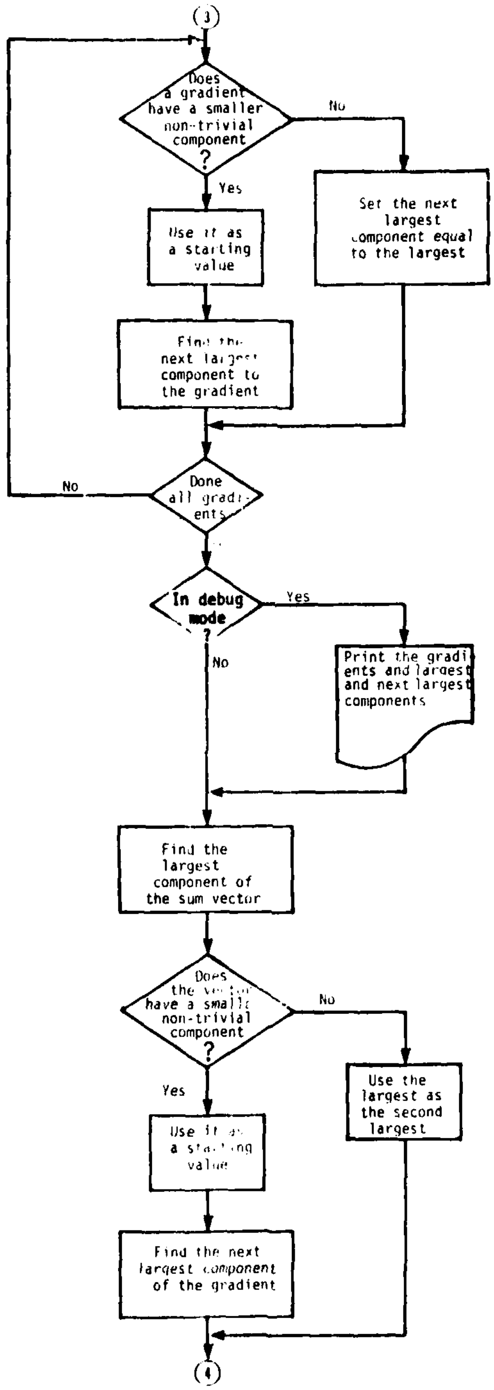
```
      DO 1700 V=1,NOVARS
      PRINT(V)=0.0
      IF ( RMINS(G) .GE. EPSLON ) PRINT(V)=GRAD(G,V)/RMINS(G)
1700  CONTINUE
      WRITE(OUTFIL,FMTWRT) (PRINT(V),V=1,NOVARS)
1800  CONTINUE
C
C PRINT THE EQUIVALENT CHANGES
C
      WRITE(OUTFIL,10100)
10100 FORMAT(//,1X,30(1H*),'EQUIVALENT CHANGES',30(1H*),/)
      DO 2000 G=1,NOGRO
      DO 1900 V=1,NOVARS
      PRINT(V)=0.0
      IF ( GRAD(G,V) .GE. EPSLON ) PRINT(V)=1./GRAD(G,V)
1900  CONTINUE
      WRITE(OUTFIL,FMTWRT) (PRINT(V),V=1,NOVARS)
2000  CONTINUE
C
C PRINT THE COMPARISONS
C
      IF (NOVARS .LT. 2) GOTO 2250
      WRITE(OUTFIL,10200)
10200 FORMAT(//,1X,30(1H*),'COMPARISONS',30(1H*),/)
      DO 2200 G=1,NOGRO
      DO 2100 V=1,NOVARS
      PRINT(V)=0.0
      IF ( RMAXS(G,1) .GE. EPSLON ) PRINT(V)=GRAD(G,V)/RMAXS(G,1)
      IF (ABS(PRINT(V)) .EQ. 1.) PRINT(V)=GRAD(G,V)/RMAXS(G,2)
2100  CONTINUE
      WRITE(OUTFIL,FMTWRT) (PRINT(V),V=1,NOVARS)
2200  CONTINUE
2250  CONTINUE
C
C SUMMED GRADIENT COMPUTATIONS
C
      WRITE(OUTFIL,10300)
10300 FORMAT(//,1X,30(1H+),'SUMMED GRADIENT COMPUTATIONS',30(1H+),/)
C
C WRITE OUT THE SUMMED GRADIENTS
C
      WRITE(OUTFIL,10350)
10350 FORMAT(//,1X,30(1H*),'SUMMED GRADIENTS',30(1H*),/)
      WRITE(OUTFIL,FMTWRT) (CSUMS(V),V=1,NOVARS)
C
C COMPUTE THE SUMMED RATIOS
C
```

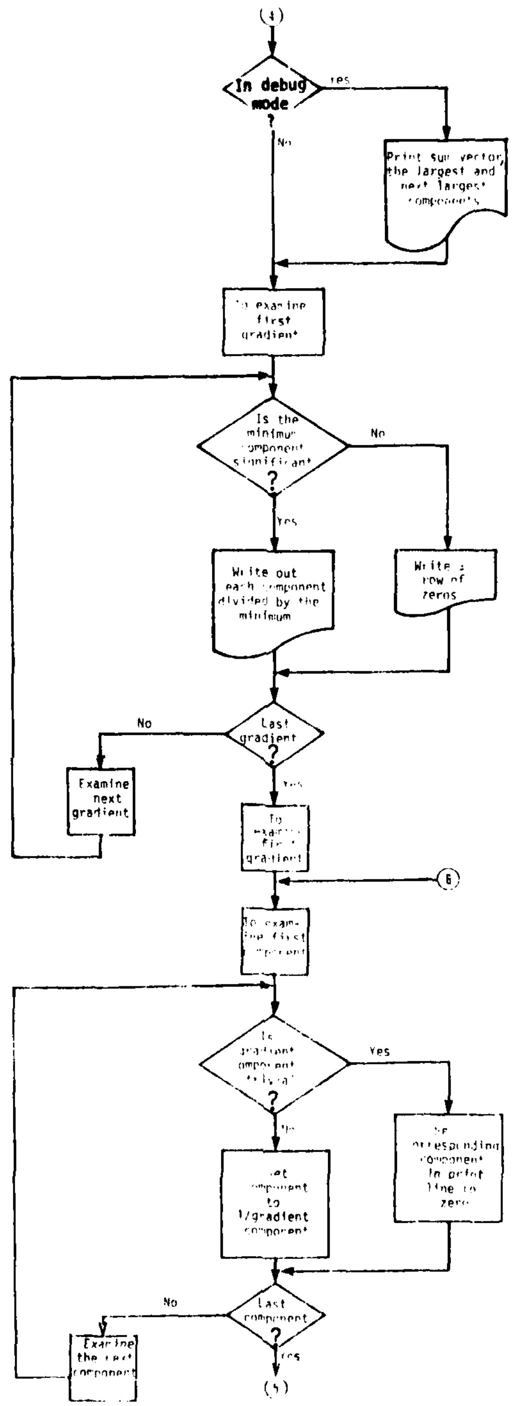
```
      WRITE(OUTFIL,10400)
10400 FORMAT(//,1X,30(1H*),'RELATIVE SUMMED RATIOS',30(1H*),/)
      DO 10500 V=1,NOVARS
      PRINT(V)=0.0
      IF ( CSMIN .GE. EPSLON ) PRINT(V)=CSUMS(V)/CSMIN
10500 CONTINUE
      WRITE(OUTFIL,FMTWRT) (PRINT(V),V=1,NOVARS)
C
C COMPUTE THE SUMMED EQUIVALENT CHANCES
C
      WRITE(OUTFIL,10600)
10600 FORMAT(//,1X,30(1H*),'EQUIVALENT SUMMED CHANCES',30(1H*),/)
      DO 10700 V=1,NOVARS
      PRINT(V)=0.0
      IF ( CSUMS(V) .GE. EPSLON ) PRINT(V)=NOGPD/CSUMS(V)
10700 CONTINUE
      WRITE(OUTFIL,FMTWRT) (PRINT(V),V=1,NOVARS)
C
C COMPUTE SUMMED COMPARISONS
C
      IF ( NOVARS .LT. 2) GOTO 11000
      WRITE(OUTFIL,10800)
10800 FORMAT(//,1X,30(1H*),'SUMMED COMPARISONS',30(1H*),/)
      DO 10900 V=1,NOVARS
      PRINT(V)=0.0
      IF ( CSMAXS(1) .GE. EPSLON ) PRINT(V)=CSUMS(V)/CSMAXS(1)
      IF (ABS(PRINT(V)) .EQ. 1.) PRINT(V)=CSUMS(V)/CSMAXS(2)
10900 CONTINUE
      WRITE(OUTFIL,FMTWRT) (PRINT(V),V=1,NOVARS)
11000 CONTINUE
      WRITE(OUTFIL,11100)
11100 FORMAT(///)
      END
```

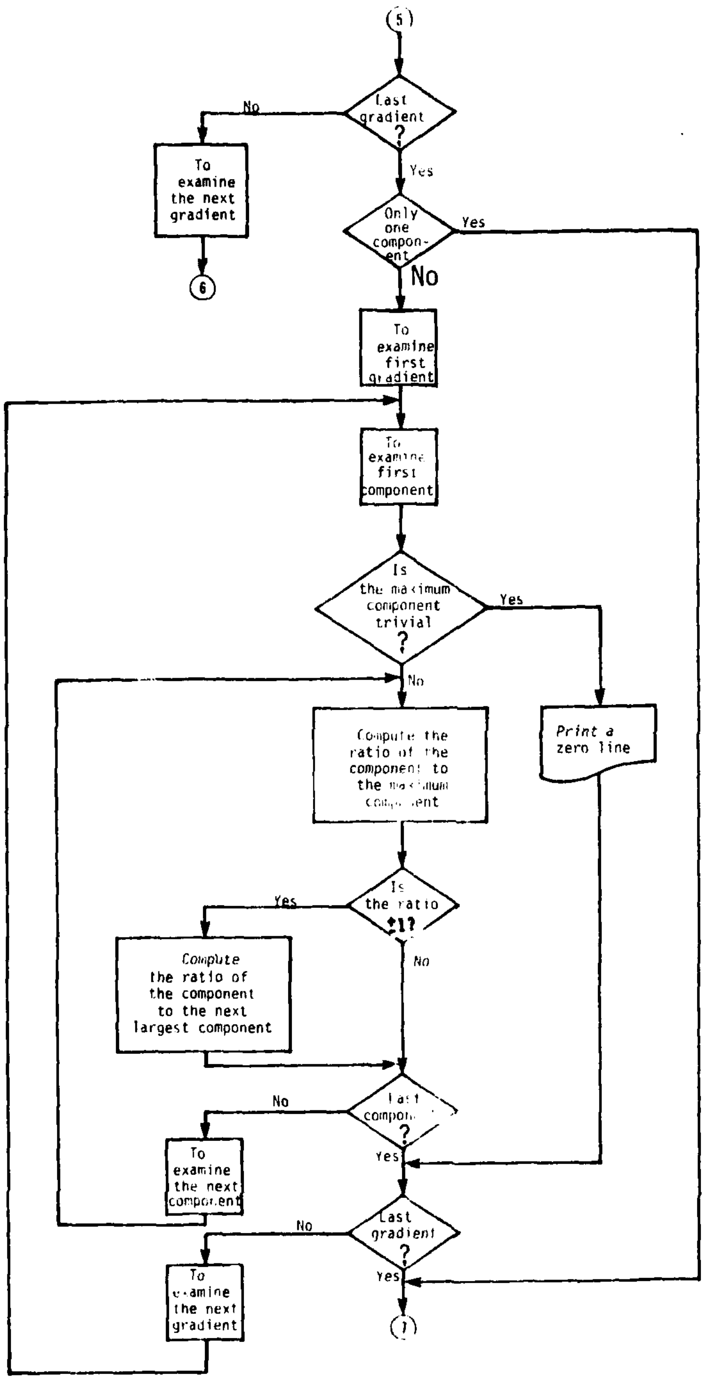
3-7. ANALYZ ROUTINE FLOWCHART

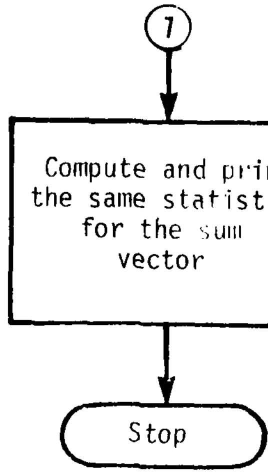



1
2
Print out the gradients
Find the first non-trivial element
Set small values to zero
Find the smallest non-trivial component for each gradient
In debug mode ?
Yes
No
Print the smallest component
Compute the vector sum of the gradients
Set small values to zero
Find the smallest non trivial component of the sum vector
Find the largest component of each gradient
3


3
Does a gradient have a smaller non-trivial component ?
No
Yes
Use it as a starting value
Set the next largest component equal to the largest
Find the next largest component to the gradient
Done all gradients
No
In debug mode ?
Yes
No
Print the gradients and largest and next largest components
Find the largest component of the sum vector
Does the vector have a smaller non-trivial component ?
No
Yes
Use the largest as the second largest
Find the next largest component of the gradient
4


(4)
In debug mode ?
Yes
No
Print sum vector, the largest and next largest components
first gradient
Is the minimum component significant ?
No
Yes
Write out each component divided by the minimum
row of zeros
Last gradient ?
No
Examine next gradient
gradient
6
Yes
Last component ?
No
Examine the next component
Yes
(5)


5
Last gradient ?
No
To examine the next gradient
6
Yes
Only one compon-ent
Yes
No
To examine first gradient
To examine first component
Is the maximum component trivial ?
Yes
No
Print a zero line
Compute the ratio of the component to the maximum component
Is the ratio
Yes
No
Compute the ratio of the component to the next largest component
Last component ?
No
To examine the next component
Yes
Last gradient ?
No
To examine the next gradient
Yes
7


7
Compute and print the same statistics for the sum vector
Stop

CHAPTER 4

THE GRID SUBSYSTEM

Section I. THE GRID ROUTINE

4-1. INTRODUCTION. This routine utilizes a user specified subdivision of each variable in order to generate a grid of input variable values. The routine to be tested is evaluated at each node of the grid, and the gradients are also computed at each node. The output of the grid routine is used by REARRANGE and DIFFQUOT.

4-2. LIMITATIONS

a. As currently compiled, the routine is restricted to 20 input variables at the most.

b. All input variables (being varied) must be real.

c. Only one output from the routine being tested may be checked out at one time.

d. Testing time-consuming Monte Carlo routines may be too expensive. Rather than making several runs per point and averaging the output values, it is better to run one component of the system at a time, testing random variables no differently from deterministic variables.

e. The function represented by the routine being tested must be well-behaved.

4-3. RUN SETUPS

a. <u>To Develop an Absolute ASCI Program File</u>

```
@MAP,S name to be given to absolute element

IN 03PROGTEST.GRID.

IN 03PROGTEST.PARTIAL.

IN element containing the driver PREPR.

IN programs to be tested.

LIB$*FTN8.

END.
```

b. To Execute

@USE 10, name of file into which points will be stored.

@ASG,A name of this file.

@USE 11, name of file into which gradients will be stored.

@ASG,A name of this file.

@USE 12, name of scratch file (used as input to REARRANGE and DIFFQUOT).

@ASG,A name of this file.

@USE 13, name of scratch file.

@ASG,A name of this file.

@XQT name of absolute deck created by @MAP.

[Input deck]

c. Description of the Input Deck

(1) Line 1. Number of variables - I3

Zero approximator - F10.7

Debug mode field - I3

The zero approximator is a threshold: numbers smaller in absolute value are considered to be zero. If the debug mode field contains the number -1, the run will be in debug mode. Any other value implies run will not print debugging information.

(2) Line 2. (Format for reading in one line of variable variation data.)

(3) Line 3. Initial variable values. Variable one's initial value is leftmost, variable two's initial value is to the right of variable one, etc.

(4) Line 4. Bounds for variable values.

(5) Line 5. Increment (step) value for each variable.

(6) Line 6. Format for one line of output. First field should be I6, other fields should be real. The number of fields specified must be no less than number of variables + 1.

(7) Line 7. File I/O format for one line of output. All fields should be real and at least (number of variables + 2) fields must be specified.

(8) Line 8. Optional; should contain a format for writing one line of debugging data if in debug mode.

d. Sample Input Data

4 .001 +1	Four variables. Zero approximator value of .001. Not in debug mode
(13F6.0)	Format for reading variable variation data
5.0 0.5 0.5 0.5	Initial values
18.00 9.00 48.00 0.9	Terminating values for incrementation
3.5 1.5 10.0 0.1	Step values for each variable
(' POINT NUMBER ',I6,5(7X,F13.5I))	Format for output
(6(7X,F13.5))	File I/O format

Not a debugging run, so debugging format is omitted.

e. Sample Run Setup

```
@USE 10,03MAT5.

@ASG,A 03MAT5.

@USE 11,03MAT6.

@ASG,A 03MAT6.

@USE 12,03MAT7.

@ASG,A 03MAT7.
```

```
@USE 13,03MAT8.

@ASG,A 03MAT8.

@XQT 03PROGTEST.COMGRID1.

  4      .001 +1

(13F6.0)
   5.0    0.5   0.5   0.5

 18.00   9.00 48.00   0.9

   3.5    1.5  10.0   0.1

(' POINT NUMBER ',I6,5(7X,F13.5))

(6(7X,F13.5))
```

4-4. OUTPUT DESCRIPTIONS AND SAMPLE OUTPUT. These outputs are of the same types as the outputs of POINTCOMP and are comprised of variable values and gradients. The two routines differ in the point selection algorithm.

a. Sample Output--Variable Values and Output Values

VARIABLE NUMBER 3 HAS BEEN INCREMENTED

POINT NUMBER	31	5.00000	.50000	50.50000	.50000	34.26218
POINT NUMBER	32	5.00000	.50000	50.50000	.60000	33.99982
POINT NUMBER	33	5.00000	.50000	50.50000	.70000	33.73710
POINT NUMBER	34	5.00000	.50000	50.50000	.80000	33.47402
POINT NUMBER	35	5.00000	.50000	50.50000	.90000	33.21060
POINT NUMBER	36	5.00000	.50000	50.50000	1.00000	32.94681

VARIABLE NUMBER 2 HAS BEEN INCREMENTED

POINT NUMBER	37	5.00000	2.00000	.50000	.50000	1.10039
POINT NUMBER	38	5.00000	2.00000	.50000	.60000	1.20860
POINT NUMBER	39	5.00000	2.00000	.50000	.70000	1.31727
POINT NUMBER	40	5.00000	2.00000	.50000	.80000	1.42641
POINT NUMBER	41	5.00000	2.00000	.50000	.90000	1.53502
POINT NUMBER	42	5.00000	2.00000	.50000	1.00000	1.64610

VARIABLE NUMBER 3 HAS BEEN INCREMENTED

POINT NUMBER	43	5.00000	2.00000	10.50000	.50000	7.16476
POINT NUMBER	44	5.00000	2.00000	10.50000	.60000	7.08334
POINT NUMBER	45	5.00000	2.00000	10.50000	.70000	7.00158
POINT NUMBER	46	5.00000	2.00000	10.50000	.80000	6.91946
POINT NUMBER	47	5.00000	2.00000	10.50000	.90000	6.83699
POINT NUMBER	48	5.00000	2.00000	10.50000	1.00000	6.75416
POINT NUMBER	48	5.00000	2.00000	10.50000	1.00000	6.75416

In each row, the four columns to the right of the point number contain the four input variable values, and the fifth contains the corresponding output value from the routine being tested. The first variable, as usual, is the leftmost, i.e., in the column to the right of the point number. Note that in each group, the last (fourth) variable is being varied through its range while the other variable values remain fixed. The headings explain how this group differs from the preceding group. Whenever the first, second, or third variable is incremented, the variables to its right are reset to their initial values.

b. Sample Output--Gradients

VARIABLE NUMBER 3 HAS BEEN INCREMENTED

POINT NUMBER	31	.26871	-2.38501	.67048	-2.62300
POINT NUMBER	32	.32286	-2.86548	.66454	-2.62653
POINT NUMBER	33	.37827	-3.34741	.65860	-2.63007
POINT NUMBER	34	.43280	-3.83228	.65264	-2.63362
POINT NUMBER	35	.48479	-4.31738	.64668	-2.63718
POINT NUMBER	36	.54242	-4.80347	.64071	-2.64073

VARIABLE NUMBER 2 HAS BEEN INCREMENTED

POINT NUMBER	37	.02158	.18565	.60644	1.08123
POINT NUMBER	38	.02576	.22479	.58747	1.08586
POINT NUMBER	39	.03090	.26340	.56843	1.09051
POINT NUMBER	40	.03513	.30235	.54930	1.09520
POINT NUMBER	41	.03931	.34164	.53010	1.09991
POINT NUMBER	42	.04344	.38127	.51081	1.10466

VARIABLE NUMBER 3 HAS BEEN INCREMENTED

POINT NUMBER	43	.15683	-.14934	.60644	-.81352
POINT NUMBER	44	.18874	-.17997	.58747	-.81700
POINT NUMBER	45	.22084	-.21085	.56843	-.82051
POINT NUMBER	46	.25426	-.24200	.54930	-.82403
POINT NUMBER	47	.28688	-.27446	.53010	-.82758
POINT NUMBER	48	.31969	-.30626	.51081	-.83115

The point numbers link the gradients to the variable values at which the gradients were computed.

c. Sample Output Files. On unit 10--points in standard format.

```
 1260  4   .001000
(6(7X,F13.5))
   5.00000        .50000       .50000       .50000       .73821
   5.00000        .50000       .50000       .60000       .77276
   5.00000        .50000       .50000       .70000       .80735
   5.00000        .50000       .50000       .80000       .84199
   5.00000        .50000       .50000       .90000       .87667
   5.00000        .50000       .50000      1.00000       .91140
   5.00000        .50000     10.50000       .50000      7.44301
   5.00000        .50000     10.50000       .60000      7.41817
   5.00000        .50000     10.50000       .70000      7.39330
```

The first line is the standard heading, component of:

(1) The number of points in I5 format.

(2) The number of variables in I3 format.

(3) The zero approximator in F10.7 format.

The second line is the format used to write/read the file. The following lines comprise the variable values and corresponding output values.

d. On Unit 11--Gradients in Standard Format

```
 1260  4   .0010000
(6(7X,F13.5))
     .00048          .30919          .67048          .34638
     .00055          .37265          .66454          .34684
     .00061          .43536          .65859          .34731
     .00067          .49823          .65264          .34778
     .00072          .56128          .64668          .34825
     .00076          .62541          .64071          .34872
     .05325         -.22734          .67048         -.24905
     .06396         -.27318          .66454         -.24938
     .07469         -.31915          .65859         -.24972
```

The first two lines are as described previously. The following lines contain gradient values corresponding to the variable values contained in the previously described file.

e. On Unit 12--File Used to Communicate with DIFFQUOT and REARRANGE

5.00000	.50000	.50000	.50000	.73821	4.00100
5.00000	.50000	.50000	.60000	.77276	4.00100
5.00000	.50000	.50000	.70000	.80735	4.00100
5.00000	.50000	.50000	.80000	.84199	4.00100
5.00000	.50000	.50000	.90000	.87667	4.00100
5.00000	.50000	.50000	1.00000	.91140	4.00100
5.00000	.50000	10.50000	.50000	7.44301	3.00100
5.00000	.50000	10.50000	.60000	7.41817	4.00100
5.00000	.50000	10.50000	.70000	7.39330	4.00100

In each row, the first four columns represent variable values. The fifth column contains output values. The integer portion of the sixth number is the variable being varied in obtaining the values for that row. The rightmost column is used by REARRANGE and DIFFQUOT. This file is written and may be read by the format located at the second line of the standard format files on units 10 and 11.

f. On Unit 13--Scratch File Containing Gradients

.00048	.30919	.67048	.34638	.73821
.00055	.37265	.66454	.34684	.77276
.00061	.43536	.65859	.34731	.80735
.00067	.49823	.65264	.34778	.84199
.00072	.56128	.64668	.34825	.87667
.00076	.62451	.64071	.34872	.91140
.05325	-.22734	.67048	-.24905	7.44301
.06396	-.27318	.66454	-.24938	7.41817
.07469	-.31915	.65859	-.24972	7.39330

The first four columns contain the gradient components. The fifth column contains the variable values corresponding to the input variable values at which the gradient was evaluated. This file was also written and may be read using the format on the second line of the files on units 10 and 11.

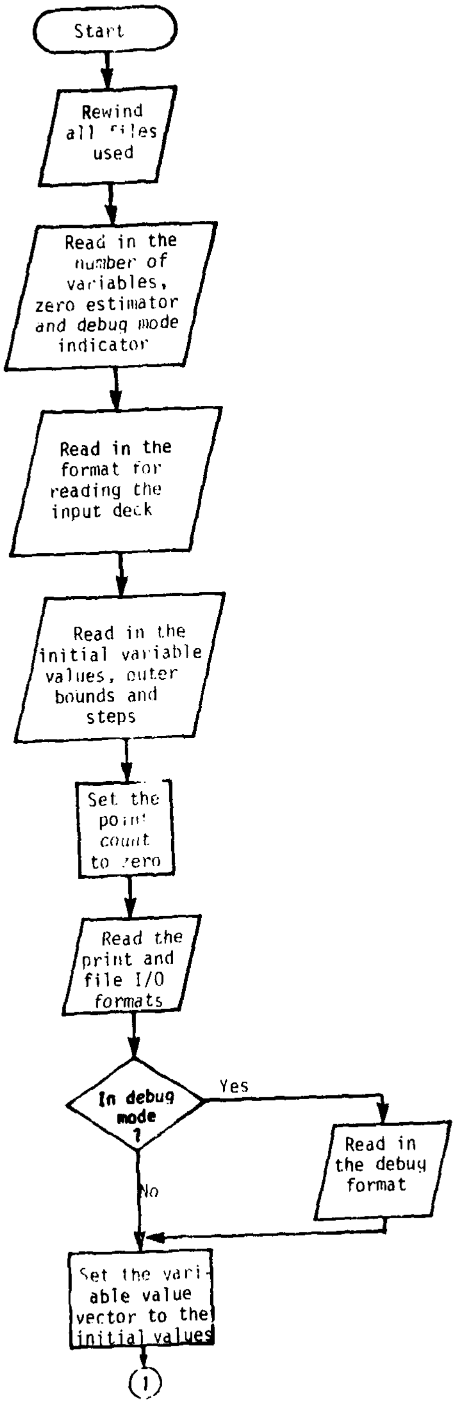
4-5. GRID ROUTINE LISTING

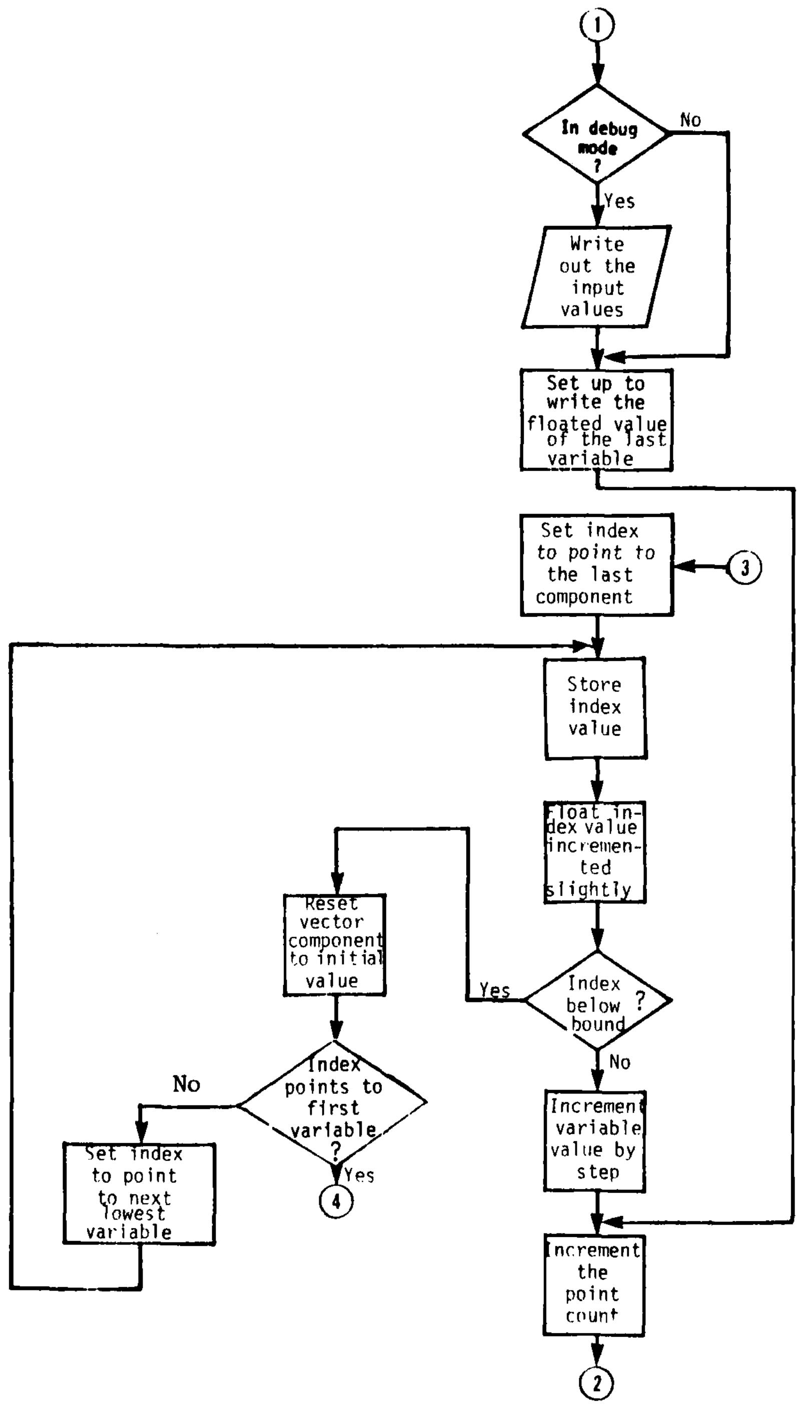
```
      PARAMETER INFILE=5,OUTFIL=6,FILE1=10,FILE2=11,FILE3=12,FILE4=13
      PARAMETER NOVALS=20,NOVAL1=NOVALS+1,NOVAL2=NOVAL1+1
C
      DIMENSION FSTVAL(NOVALS),LSTVAL(NOVALS),STEPS(NOVALS),VALU(NOVALS)
      DIMENSION COUNTS(NOVALS),LINE(NOVAL2),GRAD(NOVALS)
C
      INTEGER NOVARS,SWITCH,POINTS,COUNTS
      INTEGER PTR,PTR1,V,P
C
      REAL FSTVAL,LSTVAL,STEPS,VALU,GRAD,VALUE,LINE
      REAL EPSLON,PARTYL,INCR,FPTR
C
      CHARACTER*80 FMTRD,FMTPRT,FMTFIL,FMTDBG
C
C INITIALIZATION
C
      REWIND FILE1
      REWIND FILE2
      REWIND FILE3
      REWIND FILE4
      READ(INFILE,10000) NOVARS,EPSLON,SWITCH
10000 FORMAT(I3,F10.7,I3)
      READ(INFILE,10010) FMTRD
10010 FORMAT(A80)
      READ(INFILE,FMTRD) (FSTVAL(V),V=1,NOVARS)
      READ(INFILE,FMTRD) (LSTVAL(V),V=1,NOVARS)
      READ(INFILE,FMTRD) (STEPS(V),V=1,NOVARS)
      POINTS=0
      READ(INFILE,10010) FMTPRT
      READ(INFILE,10010) FMTFIL
      IF (SWITCH .EQ. -1) READ(INFILE,10010) FMTDBG
      DO 100 V=1,NOVARS
      VALU(V)=FSTVAL(V)
100   CONTINUE
C **********
      IF (SWITCH .NE. -1) GOTO 20000
      WRITE(OUTFIL,10020)
10020 FORMAT(/,1X,20(1H-),'ECHO PRINTOUTS',20(1H-),/)
      WRITE(OUTFIL,10025) NOVARS,EPSLON,SWITCH
10025 FORMAT(1X,I3,F10.7,I3)
      WRITE(OUTFIL,10030)
10030 FORMAT(/,1X,'** FIRST VALUES **')
      WRITE(OUTFIL,FMTDBG) (FSTVAL(V),V=1,NOVARS)
      WRITE(OUTFIL,10040)
10040 FORMAT(/,1X,'** LAST VALUES **')
      WRITE(OUTFIL,FMTDBG) (LSTVAL(V),V=1,NOVARS)
      WRITE(OUTFIL,10050)
10050 FORMAT(/,1X,'** INCREMENTS **')
```

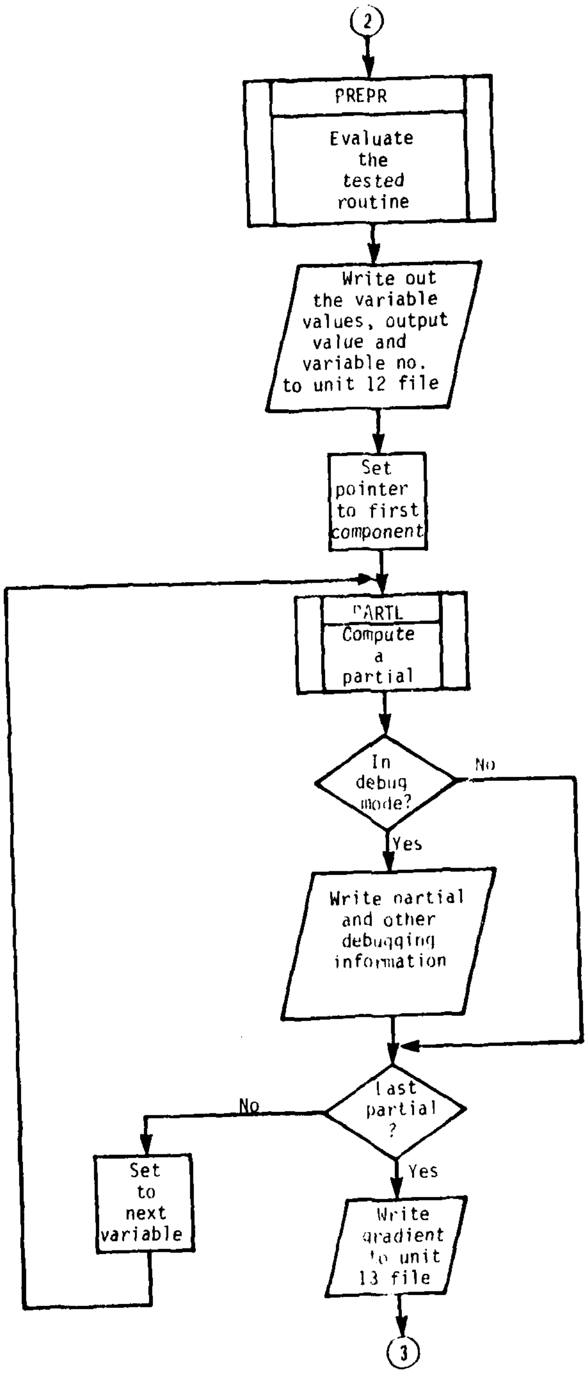
```
      WRITE(OUTFIL,FMTDBG) (STEPS(V),V=1,NOVARS)
20000 CONTINUE
C **********
      FPTR=FLOAT(NOVARS)+EPSLON
      GOTO 500
C
C TO COMPUTE THE NEXT SET OF VARIABLE VALUES
C
200   DO 300 PTR=NOVARS,1,-1
      PTR1=PTR
      FPTR=FLOAT(PTR1)+EPSLON
      IF (VALU(PTR) .LT. LSTVAL(PTR)) GOTO 400
      VALU(PTR)=FSTVAL(PTR)
300   CONTINUE
      GOTO 600
400   VALU(PTR1)=VALU(PTR1)+STEPS(PTR1)
500   CONTINUE
      POINTS=POINTS+1
      CALL PREPR(VALU,VALUE)
      WRITE(FILE3,FMTFIL) (VALU(V),V=1,NOVARS),VALUE,FPTR
      DO 550 V=1,NOVARS
      CALL PARTL(NOVARS,V,VALU,EPSLON,PARTYL,INCR,VALUE)
      GRAD(V)=PARTYL
C **********
      IF (SWITCH .NE. -1) GOTO 20010
      WRITE(OUTFIL,10060) POINTS,V
10060 FORMAT(/,1X,'** PARTIAL SUBROUTINE OUTPUT FOR POINT  ',
     1  I6,' AND VARIABLE  ',I3,4H  **)
      WRITE(OUTFIL,FMTDBG) PARTYL,VALUE,INCR
20010 CONTINUE
C **********
550   CONTINUE
      WRITE(FILE4,FMTFIL) (GRAD(V),V=1,NOVARS),VALUE
C RETURN TO COMPUTE THE NEXT SET OF VARIABLE VALUES
      GOTO 200
600   REWIND FILE3
      REWIND FILE4
      WRITE(FILE1,10070) POINTS,NOVARS,EPSLON
10070 FORMAT(I5,I3,F10.7)
      WRITE(FILE1,10010) FMTFIL
      WRITE(FILE2,10070) POINTS,NOVARS,EPSLON
      WRITE(FILE2,10010) FMTFIL
      WRITE(OUTFIL,10075) POINTS
10075 FORMAT(///,31X,'THE ',I5,' POINTS AND OUTPUT VALUES',/)
      DO 800 P=1,POINTS
      READ(FILE3,FMTFIL) (LINE(V),V=1,NOVARS),LINE(NOVAL1),LINE(NOVAL2)
      PTR=INT(LINE(NOVAL2))
      IF (PTR .EQ. NOVARS) GOTO 700
```

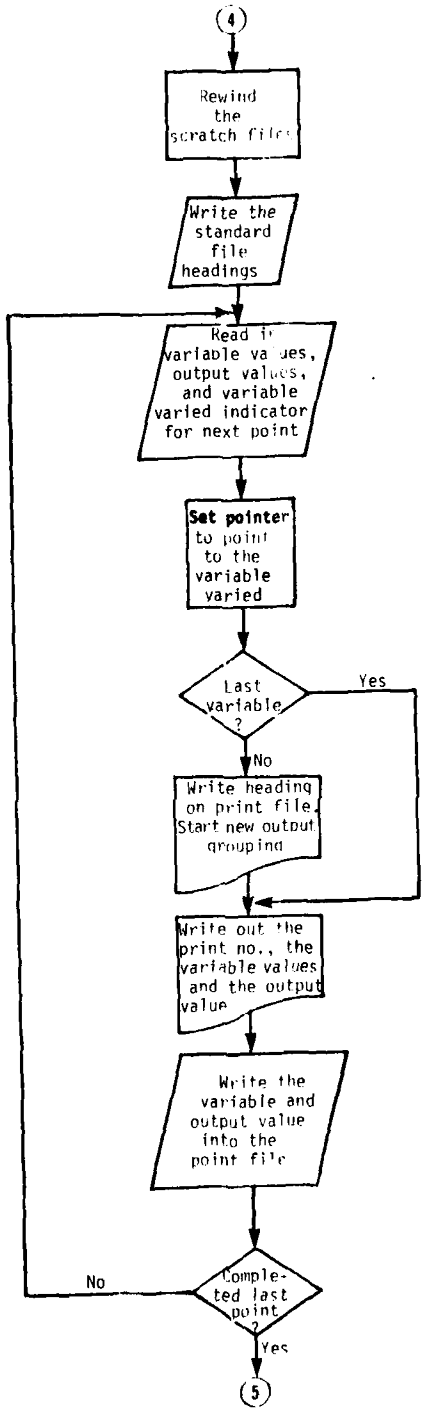
```
      WRITE(OUTFIL,10080) PTR
10030 FORMAT(/,1X,'VARIABLE NUMBER  ',I3,'  HAS BEEN INCREMENTED',
     1  /)
700   WRITE(OUTFIL,FMTPRT) P,(LINE(V),V=1,NOVARS),LINE(NOVAL1)
      WRITE(FILE1,FMTFIL) (LINE(V),V=1,NOVARS),LINE(NOVAL1)
800   CONTINUE
      WRITE(OUTFIL,10085)
10085 FORMAT(///,31X,'THE CORRESPONDING GRADIENTS',/)
      REWINC FILE3
      DO 900 P=1,POINTS
      READ(FILE3,FMTFIL) (LINE(V),V=1,NOVARS),LINE(NOVAL1),LINE(NOVAL2)
      PTR=INT(LINE(NOVAL2))
      IF (PTR .EG. NOVARS) GOTO 850
      WRITE(OUTFIL,10080) PTR
850   READ(FILE4,FMTFIL) (LINE(V),V=1,NOVARS)
      WRITE(OUTFIL,FMTPRT) P,(LINE(V),V=1,NOVARS)
      WRITE(FILE2,FMTFIL) (LINE(V),V=1,NOVARS)
900   CONTINUE
      WRITE(OUTFIL,10090)
10090 FORMAT(///)
      END
```

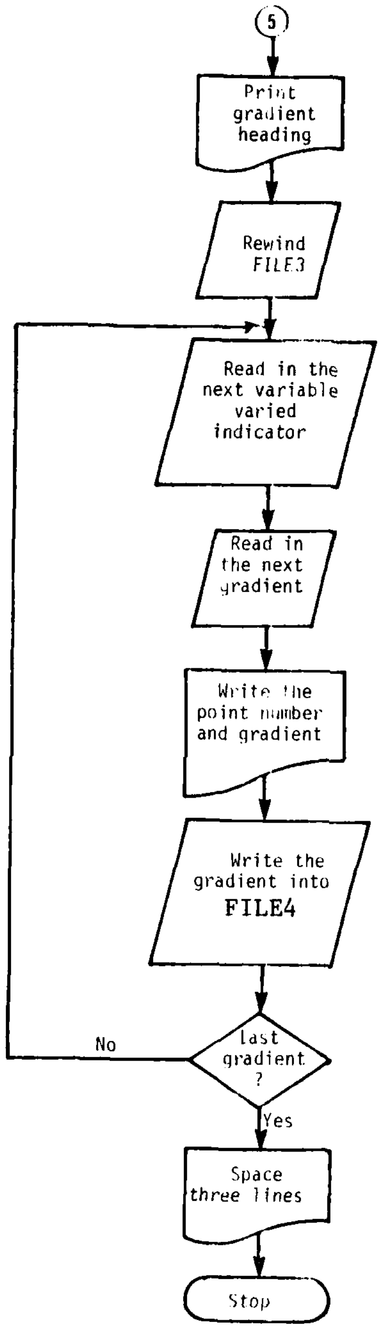
4-6. GRID ROUTINE FLOWCHART




1
In debug mode ?
No
Yes
Write out the input values
Set up to write the floated value of the last variable
Set index to point to the last component
3
Store index value
Float index value incremented slightly
Reset vector component to initial value
Yes
Index below bound ?
No
Index points to first variable ?
No
Set index to point to next lowest variable
Yes
4
Increment variable value by step
Increment the point count
2


2
PREPR
Evaluate the tested routine
Write out the variable values, output value and variable no. to unit 12 file
Set pointer to first component
PARTL
Compute a partial
In debug mode?
No
Yes
Write partial and other debugging information
Last partial ?
No
Yes
Set to next variable
Write gradient to unit 13 file
3


4
Rewind the scratch file
Write the standard file headings
Read in variable values, output values, and variable varied indicator for next point
Set pointer to point to the variable varied
Last variable ?
Yes
No
Write heading on print file. Start new output grouping
Write out the print no., the variable values and the output value
Write the variable and output value into the point file
Comple-ted last point ?
No
Yes
5


5
Print gradient heading
Rewind FILE3
Read in the next variable varied indicator
Read in the next gradient
Write the point number and gradient
Write the gradient into FILE4
last gradient ?
No
Yes
Space three lines
Stop

Section II. THE REARRANGE ROUTINE

4-7. INTRODUCTION. The output of the GRID routine is so organized that within each group of output lines, the lines agree in all variable values except the last, i.e., the last variable is varied most frequently. The REARRANGE routine reorders the output, grouping on each variable in turn, except the last. This reordering facilitates the analysis of the output.

4-8. LIMITATIONS

a. The currently compiled version is limited to 20 variables.

b. Since this routine utilizes output from GRID, the limitations of the GRID routine are applicable.

c. This routine uses much I/O, so the smaller the number of points produced by GRID, the better.

4-9. RUN SETUPS

a. <u>To Execute</u>

@USE 10, name of standard format file produced by GRID on unit 10.

@ASG,A name of above file.

@USE 12, name of communication file produced by GRID on unit 12.

@ASG,A name of above file.

@USE 14, name of file to be rearranged. This file should be the nonstandard format file produced by GRID on units 12 (points) or 13 (gradients).

@ASG,A name of this file.

@USE 15, name of scratch file into which REARRANGE places outputs.

@ASG,A name of scratch file.

@XQT name of absolute file developed earlier.

[Input deck]

b. Description and Sample of Input Deck

(1) Line 1. (' POINT NUMBER',15,2X,7(F10.5,5X),/21X,8(F10.5,5X))

This format will bring out the point number and regrouped data.

(2) Line 2. (15,2X,7(F10.5,5X),/,21X,8(F10.5,5X))

This format will be used to printout the file of reordered data.

c. Sample Input File

(1) Unit 10

```
1260  4   .0010000
(6(7X,F13.5))
   5.00000          .50000      .50000      .50000      .73821
   5.00000          .50000      .50000      .60000      .77276
   5.00000          .50000      .50000      .70000      .80735
   5.00000          .50000      .50000      .80000      .84199
   5.00000          .50000      .50000      .90000      .87667
   5.00000          .50000      .50000     1.00000      .91140
   5.00000          .50000    10.50000      .50000     7.44301
   5.00000          .50000    10.50000      .60000     7.41817
   5.00000          .50000    10.50000      .70000     7.39330
   5.00000          .50000    10.50000      .80000     7.36839
   5.00000          .50000    10.50000      .90000     7.34346
```

This is a standard format file of variable values and an output value. This file is developed by GRID on unit 10, and is described in the GRID documentation.

(2) Unit 12

5.00000	.50000	.50000	.50000	.73821	4.00100
5.00000	.50000	.50000	.60000	.77276	4.00100
5.00000	.50000	.50000	.70000	.80735	4.00100
5.00000	.50000	.50000	.80000	.84199	4.00100
5.00000	.50000	.50000	.90000	.87667	4.00100
5.00000	.50000	.50000	1.00000	.91140	4.00100
5.00000	.50000	10.50000	.50000	7.44301	3.00100
5.00000	.50000	10.50000	.60000	7.41817	4.00100
5.00000	.50000	10.50000	.70000	7.39330	4.00100
5.00000	.50000	10.50000	.80000	7.36839	4.00100
5.00000	.50000	10.50000	.90000	7.34346	4.00100

This communication file is developed by GRID on unit 12 and is described in the GRID documentation.

(3) Unit 14. This is the file to be rearranged. It may be the GRID file produced on unit 12 described earlier or the analogous nonstandard format gradient file produced by GRID on unit 13. A portion of the gradient file follows:

.00048	.30919	.67048	.34638	.73821
.00055	.37265	.66454	.34684	.77276
.00061	.43536	.65859	.34731	.80735
.00067	.49823	.65264	.34778	.84199
.00072	.56128	.64668	.34825	.87667
.00076	.62451	.64071	.34872	.91140
.05325	-.22734	.67048	-.24905	7.44301
.06396	-.27318	.66454	-.24938	7.41817

d. Sample Run Stream

```
@USE 10,03MAT5.

@UASG,A 03MAT5.

@USE 12,03MAT7.

@ASG,A 03MAT7.

@USE 14,03MAT7.

@ASG,A 03MAT7.

@USE 15,03MAT9.

@ASG,A 03MAT9.
```

```
@XQT O3PROGTEST.REARRANGE.
(' POINT NUMBER ',I5,2X,7(F10.5,5X),/,21X,8(F10.5,5X))
(I5,2X,7(F10.5,5X),1,21X,8(F10.5,5X)).
```

4-10. OUTPUT DESCRIPTIONS AND SAMPLE OUTPUT

a. Printed Output

```
+++++++++++++++++++++++VARIABLE NUMBER 1+++++++++++++++++++++++

POINT NUMBER      1   .00048   .30919   .67048   .34638   .73821
POINT NUMBER    253   .00019   .32471   .68200   .34773   .73934
POINT NUMBER    505   .00010    33081   .68706   .34836   .73985
POINT NUMBER    757   .00006   .33495   .68991   .34871   .74014
POINT NUMBER   1009   .00004   .33762   .69173   .34894   .74033

END OF VARIATION FOR INITIAL POINT 1

POINT NUMBER      2   .00055   .37265   .66454   .34684   .77276
POINT NUMBER    254   .00022   .38998   .67838   .34892   .77406
POINT NUMBER    506   .00012   .39826   .68446   .34856   .77465
POINT NUMBER    758   .00007   .40214   .68788   .34887   .77498
POINT NUMBER   1010   .00005   .40531   .69006   .34908   .77520

END OF VARIATION FOR INITIAL POINT 2
```

This particular output file is a file of gradients, a reordered version of the gradient file produced by GRID. The gradients in the first group were evaluated at points whose values for variables 2-4 are identical. Adjacent gradients were evaluated at points whose variable 1 coordinates differ by the step value for variable 1. The gradients in the second group are similar, these gradients were also evaluated at points whose coordinates 2 through 4 are identical and where adjacent gradients were evaluated at points differing only in the variable 1 coordinate and the difference is the step size. Note that the fifth column contains the corresponding output values. The points at which these gradients were evaluated are exhibited below.

+++++++++++++++++++++++VARIABLE NUMBER 1+++++++++++++++++++++++

POINT NUMBER	1	5.00000	.50000	.50000	.50000	.73821
POINT NUMBER	253	8.50000	.50000	.50000	.50000	.73934
POINT NUMBER	505	12.00000	.50000	.50000	.50000	.73985
POINT NUMBER	757	15.50000	.50000	.50000	.50000	.74014
POINT NUMBER	1009	19.00000	.50000	.50000	.50000	.74033

END OF VARIATION FOR INITIAL POINT 1

POINT NUMBER	2	5.00000	.50000	.50000	.60000	.77276
POINT NUMBER	254	8.50000	.50000	.50000	.60000	.77406
POINT NUMBER	506	12.00000	.50000	.50000	.60000	.77465
POINT NUMBER	758	15.50000	.50000	.50000	.60000	.77498
POINT NUMBER	1010	19.00000	.50000	.50000	.60000	.77520

END OF VARIATION FOR INITIAL POINT 2

Note that within each group, only variable 1 varies. The following printout is a later segment of the same output.

FINISHED VARIABLE NUMBER 1

+++++++++++++++++++++++VARIABLE NUMBER 2+++++++++++++++++++++++

NEW INITIAL POINT

POINT NUMBER	1	.00048	.30819	.67048	.34638	.73821
POINT NUMBER	37	.02158	.18565	.60644	1.08123	1.10039
POINT NUMBER	73	.05104	.12313	.56450	1.55265	1.32998
POINT NUMBER	109	.07778	.08812	.53491	1.88027	1.48843
POINT NUMBER	145	.10199	.06543	.51291	2.12195	1.60434
POINT NUMBER	181	.12181	.05001	.49491	2.30725	1.69280
POINT NUMBER	217	.13869	.04014	.48239	2.45384	1.76252

NEW INITIAL POINT

POINT NUMBER	253	.00019	.32471	.68200	.34773	.73934
POINT NUMBER	289	.00997	.23380	.63786	1.18644	1.15472
POINT NUMBER	325	.02581	.17577	.60447	1.81117	1.46142
POINT NUMBER	361	.04306	.13756	.57832	2.29508	1.69706
POINT NUMBER	397	.05990	.10961	.55728	2.67928	1.88374
POINT NUMBER	433	.07674	.09007	.54000	2.99334	2.03525
POINT NUMBER	469	.09131	.07532	.52555	3.25328	2.16066

These gradients were evaluated on point groups where only variable 2 varied within each group. The "NEW INITIAL POINT" message indicates that variable 1 varied between points 217 and 253, so these points vary in two coordinates--hence it's time to start a new grouping. This situation will not occur for variable 1 because variable 1 is varied the least. The following printout exhibits the points at which the gradients were evaluated.

FINISHED VARIABLE NUMBER 1

+++++++++++++++++++++++VARIABLE NUMBER 2+++++++++++++++++++++++

NEW INITIAL POINT

POINT NUMBER	1	5.00000	.50000	.50000	.50000	.73821
POINT NUMBER	37	5.00000	2.00000	.50000	.50000	1.10037
POINT NUMBER	73	5.00000	3.50000	.50000	.50000	1.32998
POINT NUMBER	109	5.00000	5.00000	.50000	.50000	1.48843
POINT NUMBER	145	5.00000	6.50000	.50000	.50000	1,60434
POINT NUMBER	181	5.00000	8.00000	.50000	.50000	1.69280
POINT NUMBER	217	5.00000	9.50000	.50000	.50000	1.76252

NEW INITIAL POINT

POINT NUMBER	253	8.50000	.50000	.50000	.50000	.73934
POINT NUMBER	289	8.50000	2.00000	.50000	.50000	1.15472
POINT NUMBER	325	8.50000	3.50000	.50000	.50000	1.46142
POINT NUMBER	361	8.50000	5.00000	.50000	.50000	1.69706
POINT NUMBER	397	8,50000	6.50000	.50000	.50000	1.88374
POINT NUMBER	433	8.50000	8.00000	.50000	.50000	2.03525
POINT NUMBER	469	8.50000	9.50000	.50000	.50000	2.16066

Note that all points in each group vary only in the second variable. Note also that, as before, while the point numbers continue increasing, points 217 and 253 differ in two coordinates since variables 1 and 2 change simultaneously. This fact necessitates creating a new group.

b. Sample Output File Segment

252	5.00000	9.50000	50.50000	1.00000	15.70270
504	8.50000	9.50000	50.50000	1.00000	21.01859
756	12.00000	9.50000	50.50000	1.00000	24.55868
1008	15.50000	9.50000	50.50000	1.00000	27.08553
1260	19.00000	9.50000	50.50000	1.00000	28.97974

```
 252+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   1   ////////////////////////////////////////////////////////
```

VARIABLE 2

---------------------- NEW INITIAL POINT-------------------------

1	5.00000	.50000	.50000	.50000	.73821
37	5.00000	2.00000	.50000	.50000	1.10039
73	5.00000	3.50000	.50000	.50000	1.32998
109	5.00000	5.00000	.50000	.50000	1.48843
145	5.00000	6.50000	.50000	.50000	1.60434
181	5.00000	8.00000	.50000	.50000	1.69280
217	5.00000	9.50000	.50000	.50000	1.76252

The leftmost integers are point numbers.

(1) The line
252 + + + . . . indicates the end of the grouping whose initial point was 252, i.e., if there was a point 253, the point numbers, now at 1260, would decrease to 253 next and commence to increase from that value, i.e., the next point numbers would be:

253
289
325
etc.

(2) The line
1 //// . . . indicates the end of regroupings for which only variable 1 varies within each preceding group.

(3) The line
VARIABLE 2 indicates that now groupings where only variable 2 varies will be derived.

(4) The line
---- . . . ----NEW INITIAL POINT --- . . . indicates that while the point numbers may continue to increase, a new grouping must nevertheless begin.

4-11. REARRANGE ROUTINE LISTING

```
      PARAMETER INFILE=5,OUTFIL=6,FILE1=10,FILE3=12,SRTFIL=14,GRPFIL=15
      PARAMETER NOVALS=20,NOVAL1=NOVALS+1,NOVAL2=NOVALS+2
C
      CHARACTER*80 FMTFIL,FMTPRT,FMTGRP
C
      INTEGER POINTS,PTR,STEP,STEPN,PERIOD,F,I,Q,Q1,P,V
      INTEGER NOVRM1,NOVARS,NOVRP1,NOVRP2
C
      REAL LINE
C
      DIMENSION PERIOD(NOVALS),LINE(NOVAL2)
C
C INITIALIZATION
C
      REWIND FILE1
      REWIND FILE3
      REWIND SRTFIL
      REWIND GRPFIL
      READ(FILE1,10000) POINTS,NOVARS
10000 FORMAT(I5,I3)
      READ(FILE1,10010) FMTFIL
10010 FORMAT(A80)
      READ(INFILE,10010) FMTPRT
      READ(INFILE,10010) FMTGRP
      NOVRP1=NOVARS+1
      NOVRP2=NOVARS+2
      NOVRM1=NOVARS-1
C
C TO FIND THE PERIODS
C
      DO 100 V=1,NOVARS
      PERIOD(V)=0
100   CONTINUE
      DO 200 P=1,POINTS
      READ(FILE3,FMTFIL) (LINE(I),I=1,NOVRP2)
      PTR=INT(LINE(NOVRP2))
      IF(PERIOD(PTR) .EQ. 0) PERIOD(PTR)=MAX0(P-1,1)
      IF(PERIOD(1) .NE. 0) GOTO 300
200   CONTINUE
C
C TO PRODUCE A REORDERED FILE, VARYING EACH VARIABLE THROUGHOUT
C     ITS RANGE IN TURN
C
300   DO 600 F=1,NOVRM1
      WRITE(OUTFIL,10020) F
10020 FORMAT(//,1X,30(1H*),'VARIABLE NUMBER ',I5,2X,30(1H*),//)
      WRITE(GRPFIL,10030) F
10030 FORMAT('VARIABLE ',I5)
```

AD-A091 653 ARMY CONCEPTS ANALYSIS AGENCY BETHESDA MD F/G 9/2
PROGTEST: A COMPUTER SYSTEM FOR THE ANALYSIS OF COMPUTATIONAL C--ETC(U)
APR 80 S BRAVY

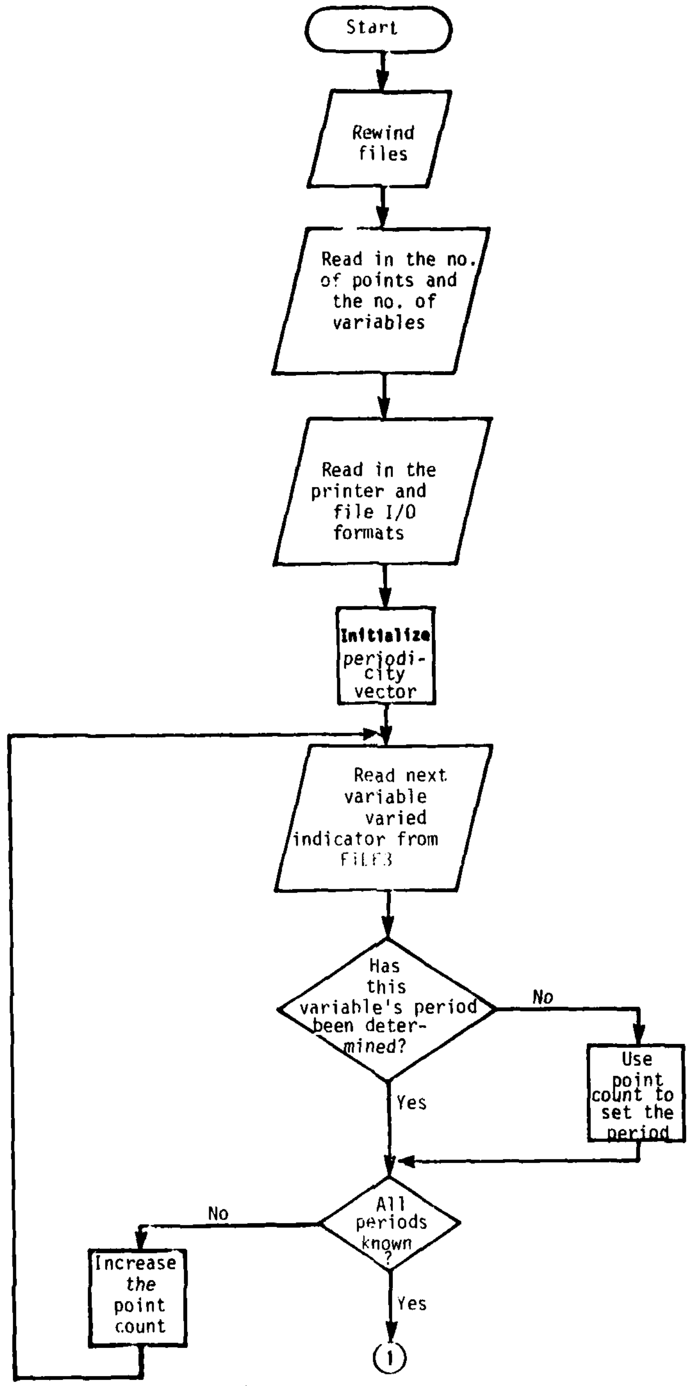
UNCLASSIFIED CAA-D-80-1 NL

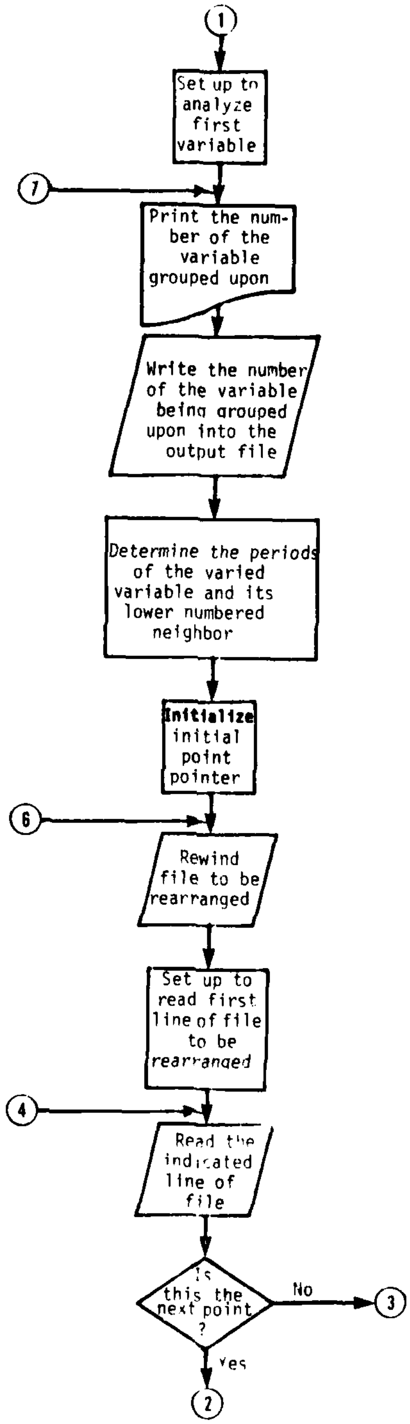
2 OF 2
AD A091653

END
DATE FILMED
12-80
DTIC

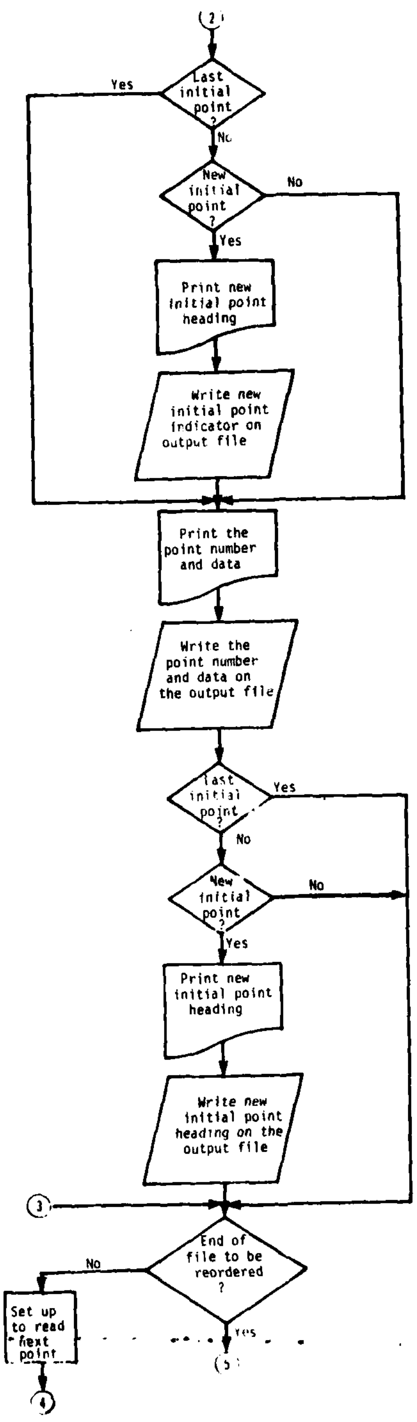
```
      STEPN=0
      STEP=PERIOD(F)
      IF (F .NE. 1) STEPN=PERIOD(F-1)
C
C TO PICK UP THE Q TH ENTRY FROM EACH BLOCK OF SIZE STEP
C
      DO 500 Q=1,STEP
      REWIND SRTFIL
      Q1=Q
      IF (Q1 .EQ. STEP) Q1=0
      DO 400 P=1,POINTS
      READ(SRTFIL,FMTFIL) (LINE(V),V=1,NOVRP1)
      IF (MOD(P,STEP) .NE. Q1) GOTO 400
      IF (Q1 .EQ. 0) GOTO 350
C
C CHECK FOR VARIATION IN THE NEXT VARIABLE
C
      IF ((F .NE. 1) .AND. (MOD(P,STEPN) .EQ. Q1)) WRITE(OUTFIL,10032)
10032 FORMAT(/,' NEW INITIAL POINT',/)
      IF ((F .NE. 1) .AND. (MOD(P,STEPN) .EQ. Q1)) WRITE(GRPFIL,10036)
10036 FORMAT(55(1H-),'NEW INITIAL POINT',55(1H-))
350   WRITE(OUTFIL,FMTPRT) P,(LINE(V),V=1,NOVARS),LINE(NOVRP1)
      WRITE(GRPFIL,FMTGRP) P,(LINE(V),V=1,NOVARS),LINE(NOVRP1)
      IF ( Q1 .NE. 0 ) GOTO 400
      IF ((F .NE. 1) .AND. (MOD(P,STEPN) .EQ. 0)) WRITE(OUTFIL,10032)
      IF ((F .NE. 1) .AND. (MOD(P,STEPN) .EQ. 0)) WRITE(GRPFIL,10036)
400   CONTINUE
      WRITE(OUTFIL,10040) Q
10040 FORMAT(/,' END OF VARIATION FOR INITIAL POINT ',I5,/)
      WRITE(GRPFIL,10050) Q
10050 FORMAT(I5,120(1H+))
C
C FINISHED PASS FOR THE Q TH ENTRY IN EACH BLOCK
C
500   CONTINUE
      WRITE(OUTFIL,10060) F
10060 FORMAT(/,' FINISHED VARIABLE NUMBER ',I3)
      WRITE(GRPFIL,10070) F
10070 FORMAT(I5,5X,120(1H/))
500   CONTINUE
      WRITE(OUTFIL,10080)
10080 FORMAT(///)
      END
```

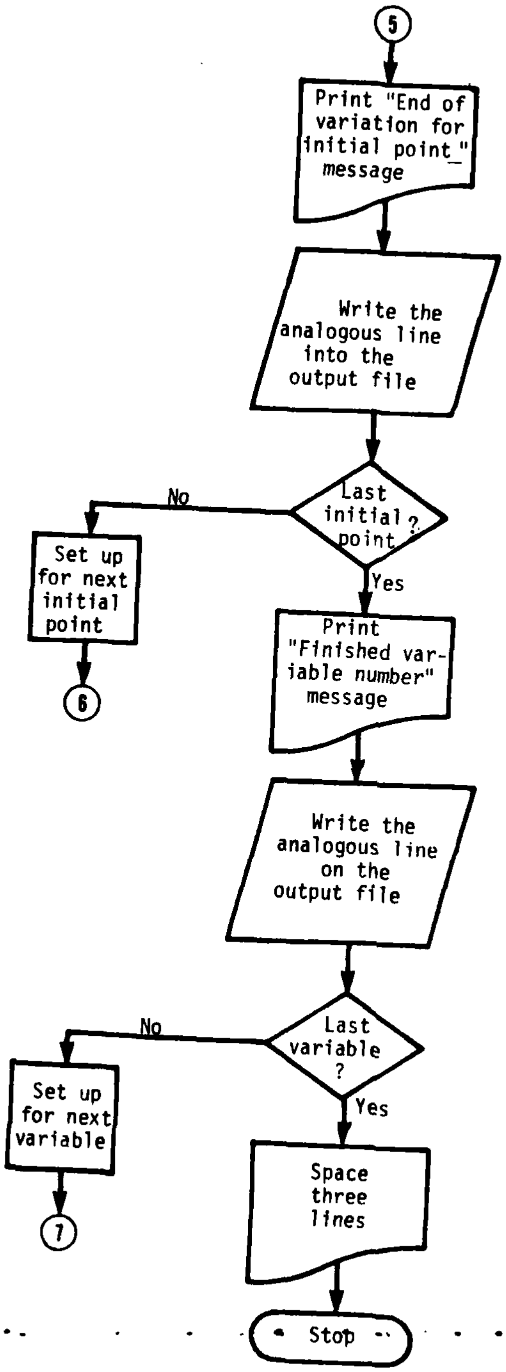
4-12. REARRANGE ROUTINE FLOWCHART





1
Set up to analyze first variable
7
Print the number of the variable grouped upon
Write the number of the variable being grouped upon into the output file
Determine the periods of the varied variable and its lower numbered neighbor
Initialize initial point pointer
6
Rewind file to be rearranged
Set up to read first line of file to be rearranged
4
Read the indicated line of file
Is this the next point ?
No
3
Yes
2


2
Last initial point ?
Yes
No
New initial point ?
No
Yes
Print new initial point heading
Write new initial point indicator on output file
Print the point number and data
Write the point number and data on the output file
Last initial point ?
Yes
No
New initial point ?
No
Yes
Print new initial point heading
Write new initial point heading on the output file
3
End of file to be reordered ?
No
Yes
Set up to read next point
4
5


5
Print "End of variation for initial point_" message
Write the analogous line into the output file
Last initial point?
No
Yes
Set up for next initial point
6
Print "Finished variable number" message
Write the analogous line on the output file
Last variable ?
No
Yes
Set up for next variable
7
Space three lines
Stop

Section III. THE DIFFQUOT ROUTINE

4-13. INTRODUCTION. This routine utilizes the output developed by GRID on unit 10 in order to compute difference quotients for one variable. The point numbers listed in the output identify the points used to compute the difference quotients. These points are developed by GRID, and the point numbers link the GRID output points to the difference quotient computations.

4-14. BACKGROUND. See the discussion of difference quotients in the VARVARY1 documentation (Chapter 5).

4-15. LIMITATIONS

a. The current compiled version is limited to 20 variables.

b. The current compiled version is limited to a maximum of 500 lines of difference quotient computations in each difference quotient block.

c. Since this routine utilizes GRID output, the limitations applicable to GRID apply.

4-16. RUN SETUPS

a. <u>To Develop an Absolute ASCI Program File</u>

@MAP,S name of absolute element.

IN O3PROGTEST.DIFFQUOT.

IN O3PROGTEST.PARTIAL.

IN element containing the driver PREPR.

IN programs to be tested.

LIB$*FTN8.

END

b. To Execute

@USE 10, name of file produced by GRID on unit 10.

@ASG,A name of above file.

@USE 12, name of file produced by GRID on unit 12.

@ASG,A name of this file.

@XQT name of absolute deck created earlier.

[Input deck]

c. Sample Input Deck and Description

(1) Line 1.

```
  2
```

The number of the variable for which difference quotients are to be computed, in I3 format.

(2) Line 2.

```
(13F6.0)
```

The format for reading each line of input data.

(3) Line 3.

```
   5.0   0.5   0.5   0.5
```

Initial values for each variable, where variable 1 is leftmost.

(4) Line 4.

```
 18.00  9.00 48.00   0.9
```

Terminal values for each variable, variable 1 leftmost.

(5) Line 5.

```
   3.5   1.5  10.0   0.1
```

Step values for each variable.

(6) Line 6.

(1X,8F13.5)

Format for one line of difference quotient printouts.

(7) Line 7.

(1X,8(5X,I6))

Format for printing out the point numbers whose difference quotients are being computed.

d. Sample Input Files and Descriptions

(1) Unit 10

```
 1260  4  .0010000
(6(7X,F13.5))
      5.00000      .50000      .50000      .50000      .73821
      5.00000      .50000      .50000      .50000      .77276
      5.00000      .50000      .50000      .70000      .89735
      5.00000      .50000      .50000      .80000      .84199
      5.00000      .50000      .50000      .90000      .87667
      5.00000      .50000      .50000     1.00000      .91140
      5.00000      .50000    10.50000      .50000     7.44301
      5.00000      .50000    10.50000      .60000     7.41817
      5.00000      .50000    10.50000      .70000     7.39330
```

This file is in standard format; it contains variable values (points) and output. This file is produced by GRID on unit 10, see GRID documentation (Chapter 4) for a more detailed description.

(2) Unit 12

```
5.00000     .50000     .50000     .50000     .73821    4.00100
5.00000     .50000     .50000     .60000     .77276    4.00100
5.00000     .50000     .50000     .70000     .80735    4.00100
5.00000     .50000     .50000     .80000     .84199    4.00100
5.00000     .50000     .50000     .90000     .87667    4.00100
5.00000     .50000     .50000    1.00000     .91140    4.00100
5.00000     .50000   10.50000     .50000    7.44301    3.00100
5.00000     .50000   10.50000     .60000    7.41817    4.00100
5.00000     .50000   10.50000     .70000    7.39330    4.00100
```

This file is produced by GRID on unit 12 and is used by DIFFQUOT to determine the periodicity of each variable. For a more detailed description, see the GRID documentation (Chapter 7).

e. <u>Sample RUN SETUP</u>

```
@USE 10,03MAT5.

@ASG,A 03MAT5.

@USE 12,03MAT7.

@ASG,A 03MAT7.

@XQT 03PROGTEST.DIFFQUOT
  1
(13F6.0)
   5.0   0.5   0.5   0.5
 18.00  9.00 48.00   0.9
   3.5   1.5  10.0   0.1
(1X,8F13.5)
(1X,8(5X,16))
```

4-17. DESCRIPTION AND SAMPLE OUTPUT

```
------------DIFFERENCE QUOTIENTS FOR VARIABLE  1------------
                       AND POINTS
```

1	253	505	757	1009
.00000				
.00032	.00000			
.00023	.00015	.00000		
.00018	.00011	.00008	.00000	
.00015	.00009	.00007	.00005	.00000

The following figure (4-1) may help explain the output:

The slope of the line to point

From point	1	253	505	757	1009
1	—	.0032	.00023	.00018	.00015
253		—	.00015	.00011	.00009
505			—	.00008	.00007
757				—	.00005
1009					—

Slope of the line from point no. 1 to point no. 253

Slope of the line from point no. 253 to point no. 757

Figure 4-1. Explanatory Figure

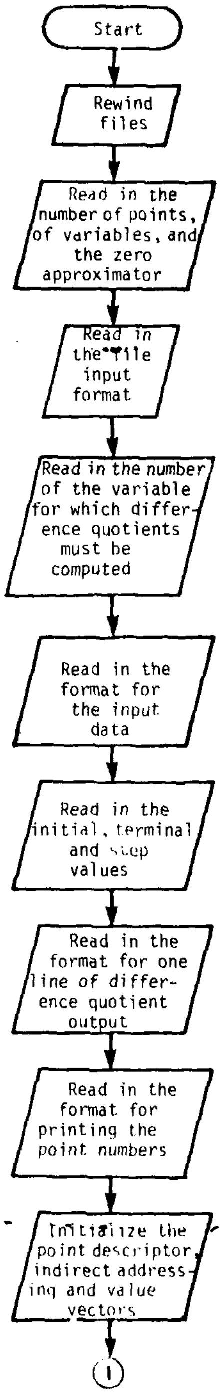
For more details, examine the description of difference quotients in the VARVARY1 documentation (Chapter 2, Section II).

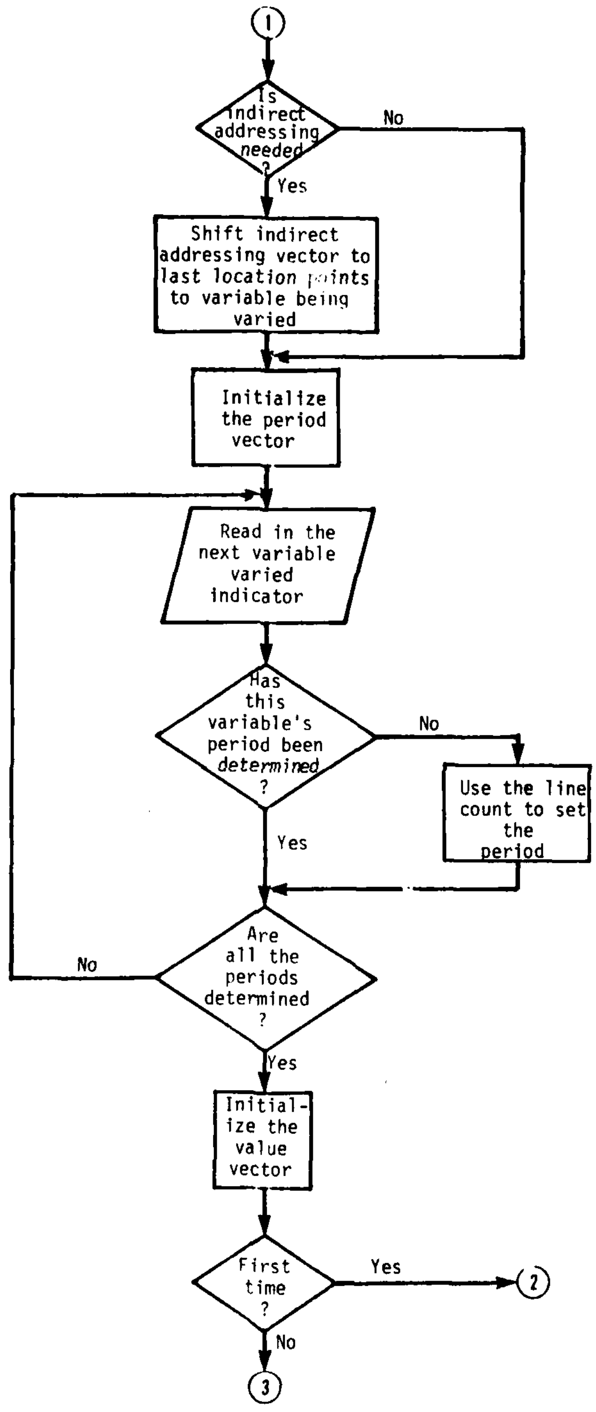
4-18. DIFFQUOT ROUTINE LISTING

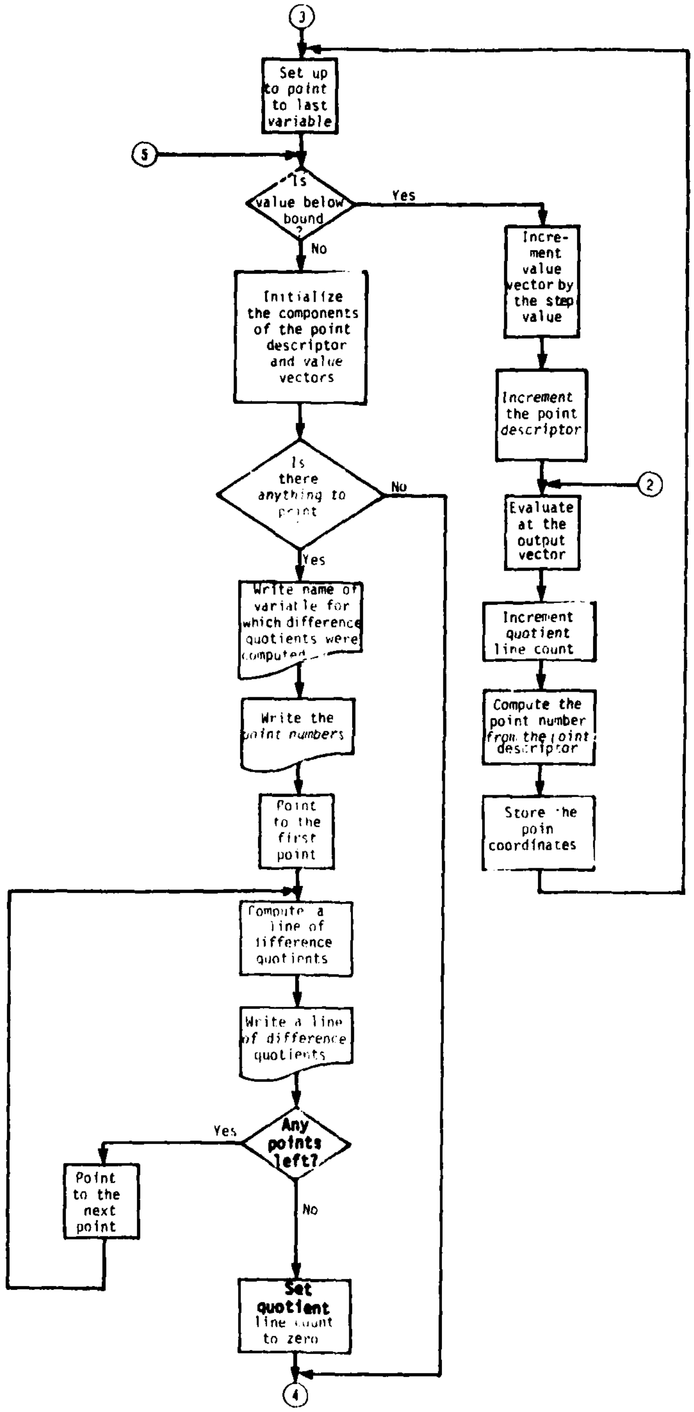
```
      PARAMETER INFILE=5,OUTFIL=6,NOPTS=500,NOVALS=20,FILE1=10
      PARAMETER FILE3=12,NOVAL1=NOVALS+1,NOVAL2=NOVAL1+1
C
      DIMENSION FSTVAL(NOVALS),LSTVAL(NOVALS),STEPS(NOVALS)
      DIMENSION INDEX(NOVALS),VALU(NOVAL2),COORDS(NOPTS,NOVAL1)
      DIMENSION DIFF(NOVALS),POINTS(NOPTS),CLOCK(NOVALS)
      DIMENSION PERIOD(NOVALS)
C
      REAL EPSLON,FSTVAL,LSTVAL,STEPS,VALU,DIFF,COORDS
C
      INTEGER INDEX,POINTS,NOVARS,NOVRM1,NOVRP1,WHICH,VAR,VAR1
      INTEGER LOC,PTCT,I,J,V,CLOCK,PERIOD,PTP,NUMBER
C
      CHARACTER*80 FMTRD,FMTPRT,FMTPR2,FMTFIL
C
      REWIND FILE1
      REWIND FILE3
      READ(FILE1,10040) NUMBER,NOVARS,EPSLON
10040 FORMAT(I5,I3,F10.7)
      READ(FILE1,10010) FMTFIL
C
      READ(INFILE,10000) WHICH
10000 FORMAT(I3)
      READ(INFILE,10010) FMTRD
10010 FORMAT(A80)
      READ(INFILE,FMTRD) (FSTVAL(V),V=1,NOVARS)
      READ(INFILE,FMTRD) (LSTVAL(V),V=1,NOVARS)
      READ(INFILE,FMTRD) (STEPS(V),V=1,NOVARS)
      READ(INFILE,10010) FMTPRT
      READ(INFILE,10010) FMTPR2
C
      NOVRP1=NOVARS+1
      NOVRP2=NOVRP1+1
      NOVRM1=NOVARS-1
      DO 100 I=1,NOVARS
      CLOCK(I)=1
      INDEX(I)=I
      VALU(I)=FSTVAL(I)
 100  CONTINUE
      IF ( WHICH .EQ. NOVARS ) GOTO 300
      DO 200 I=WHICH,NOVRM1
      INDEX(I)=INDEX(I+1)
 200  CONTINUE
      INDEX(NOVARS)=WHICH
 300  LOC=0
      DO 1200 I=1,NOVARS
      PERIOD(I)=0
 1200 CONTINUE
```

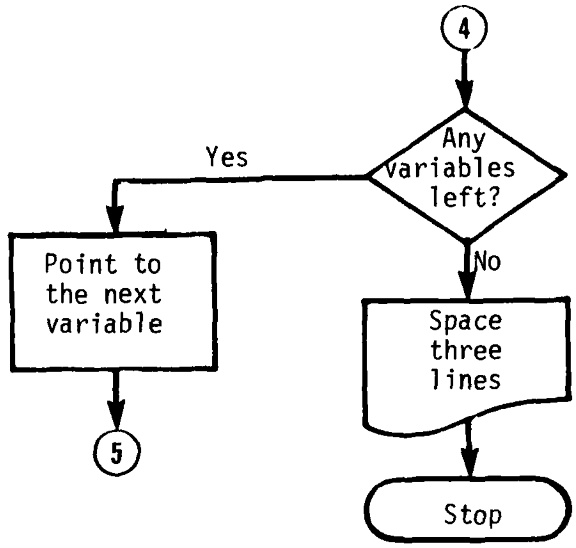
```
      DO 1300 I=1,NUMBER
      READ(FILE3,FMTFIL) (VALU(J),J=1,NOVRP2)
      PTR=INT(VALU(NOVRP2))
      IF ( PERIOD(PTR) .EQ. 0 ) PERIOD(PTR)=MAX0(I-1,1)
      IF ( PERIOD(1) .NE. 0 ) GOTO 1350
1300  CONTINUE
      STOP
1350  DO 1375 I=1,NOVARS
      VALU(I)=FSTVAL(I)
1375  CONTINUE
      GOTO 1000
300   DO 400 VAR=NOVARS,1,-1
      VAR1=VAR
      IF ( VALU(INDEX(VAR)) .LT. LSTVAL(INDEX(VAR)) ) GOTO 700
      CLOCK(INDEX(VAR))=1
      VALU(INDEX(VAR))=FSTVAL(INDEX(VAR))
C IF LOC IS ZERO, HAVE ALREADY PRINTED THE LAST SET
C    OF DIFFERENCES
      IF ( LOC .EQ. 0 ) GOTO 400
      WRITE(OUTFIL,10020) WHICH
10020 FORMAT(//,1X,20(1H-),'DIFFERENCE QUOTIENTS FOR VARIABLE ',
     1 I5,2X,20(1H-))
      WRITE(OUTFIL,10025)
10025 FORMAT(31X,'AND POINTS')
      WRITE(OUTFIL,FMTPR2) (POINTS(J),J=1,LOC)
      WRITE(OUTFIL,10028)
10028 FORMAT(/)
      DO 500 I=1,LOC
      DIFF(I)=0.
      DO 600 J=1,I
      IF ( J .EQ. I ) GOTO 600
      DIFF(J)=(COORDS(I,NOVRP1)-COORDS(J,NOVRP1))/
     1 (COORDS(I,WHICH)-COORDS(J,WHICH))
600   CONTINUE
      WRITE(OUTFIL,FMTPRT) (DIFF(J),J=1,I)
500   CONTINUE
      LOC=0
400   CONTINUE
      WRITE(OUTFIL,10030)
10030 FORMAT(///)
      STOP
700   VALU(INDEX(VAR1))=VALU(INDEX(VAR1))+STEPS(INDEX(VAR1))
      CLOCK(INDEX(VAR1))=CLOCK(INDEX(VAR1))+1
1000  CALL PREPR(VALU,VALU(NOVRP1))
      LOC=LOC+1
      POINTS(LOC)=1
      DO 1100 I=1,NOVARS
      POINTS(LOC)=POINTS(LOC)+(CLOCK(I)-1)*PERIOD(I)
1100  CONTINUE
      DO 800 I=1,NOVRP1
      COORDS(LOC,I)=VALU(I)
800   CONTINUE
      GOTO 300
      END
```

4-19. DIFFQUOT ROUTINE FLOWCHART




1
Is indirect addressing needed ?
No
Yes
Shift indirect addressing vector to last location points to variable being varied
Initialize the period vector
Read in the next variable varied indicator
Has this variable's period been determined ?
No
Use the line count to set the period
Yes
Are all the periods determined ?
No
Yes
Initial-ize the value vector
First time ?
Yes
2
No
3


3
Set up to point to last variable
5
Is value below bound?
Yes
No
Increment value vector by the step value
Increment the point descriptor
2
Evaluate at the output vector
Increment quotient line count
Compute the point number from the point descriptor
Store the point coordinates
Initialize the components of the point descriptor and value vectors
Is there anything to print
No
Yes
Write name of variable for which difference quotients were computed
Write the point numbers
Point to the first point
Compute a line of difference quotients
Write a line of difference quotients
Any points left?
Yes
No
Point to the next point
Set quotient line count to zero
4


4
Any variables left?
Yes
No
Point to the next variable
5
Space three lines
Stop

CHAPTER 5

COMMON SUBROUTINES

5-1. INTRODUCTION. Two subroutines are used by both the POINTCOMP and GRID subsystems:

a. PARTL

b. PREPR

Detailed writeups of these subroutines follow.

Section I. THE PARTL SUBROUTINE

5-2. INTRODUCTION TO THE PARTL SUBROUTINE. This routine computes the partial derivative at a point numerically by using the definition of the right partial derivative. PARTL is called by POINTCOMP and other routines repetitively to construct the gradient, but it may be used by anyone as a standalone subroutine. PARTL calls PREPR which must be a user-provided driver subroutine whose function is to obtain an output value from the program being tested.

5-3. BACKGROUND. Given a real valued function $f(x_1, \ldots, x_n)$, the jth right partial derivative of f at $(x_1, \ldots, x_n)$ is defined to be:

$$\frac{\partial f}{\partial x_j}(x_1, \ldots, x_n) = \lim_{h \to 0^+} \frac{f(x_1, \ldots, x_{j-1}, x_j+h, x_{j+1}, \ldots, x_n) - f(x_1, \ldots, x_n)}{h}$$

5-4. DISCUSSION OF METHODOLOGY

a. The methodology is derived directly from the definition:

For h = 1/2, we approximate the jth partial derivative at $(x_1, \ldots, x_n)$ to be:

$$\frac{f(x_1, \ldots, x_{j-1}, x_j+1/2, \ x_{j+1}, \ldots, x_n) - f(x_1, \ldots, x_n)}{1/2}$$

b. We repeat this procedure for h = 1/4, 1/8, 1/16, etc. When the difference between two successive approximations is sufficiently small (as determined by an input parameter), the process

is terminated and the approximation is returned as the value of the partial derivative. The process terminates automatically after 50 approximations.

5-5. LIMITATIONS

a. As currently compiled, all inputs to PARTL must be real.

b. The function whose partials are being computed should be differentiable.

5-6. CALLING SEQUENCE

Call PARTL (I1, I2, R1, R2, R3, R4, R5)

where

I1 is an integer input variable containing the number of variables.

I2 is an integer input variable containing the index value of the variable whose partial is to be found. The index value refers to the subscript locating the variable in the input array R1.

R1 is a real input array containing the values of the variables at the point at which the partial is to be evaluated.

R2 is a real input variable containing a positive number. Any number smaller than this number will be considered to be zero.

R3 is a real output variable into which the approximate value of the partial will be placed by PARTL.

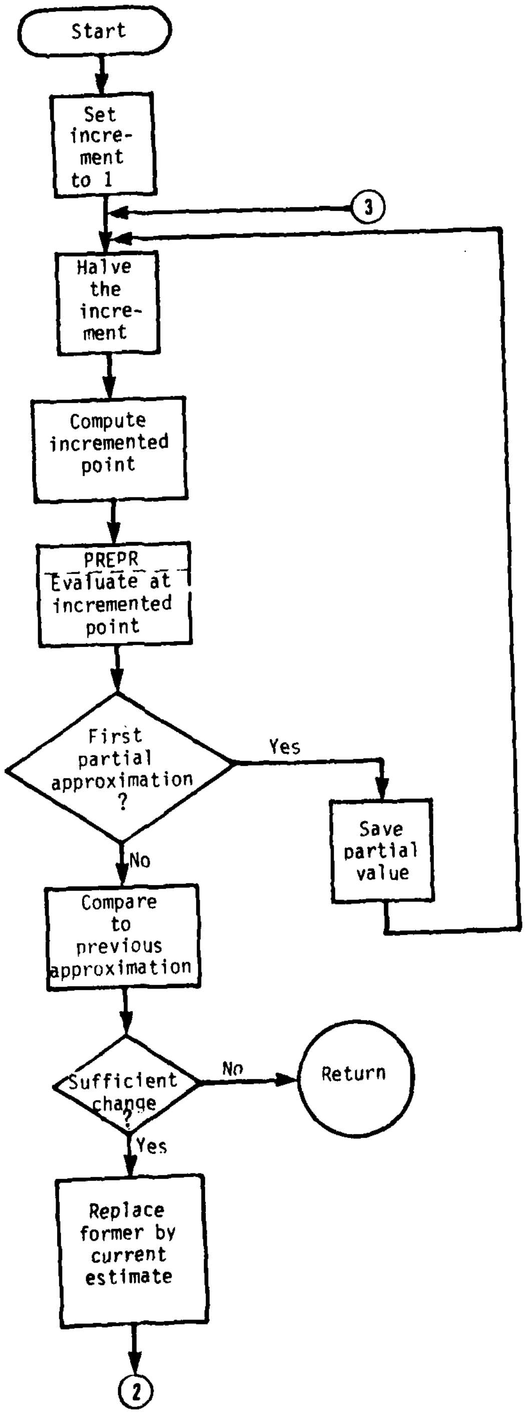
R4 A real output variable into which the last increment tested will be placed by PARTL.

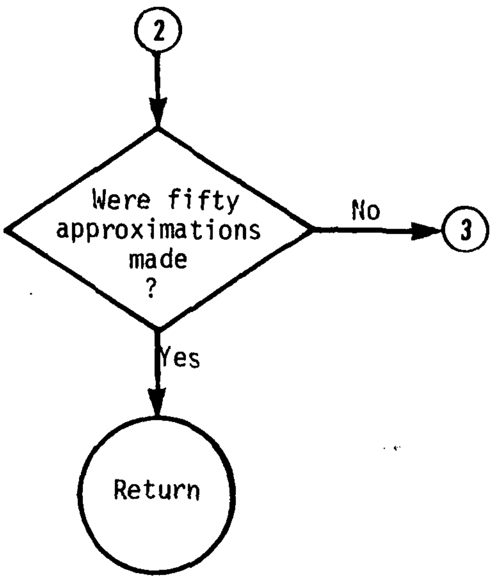
R5 is a real input variable containing the value returned by PREPR for the variable values in R1.

5-7. PARTL SUBROUTINE LISTING

```
      SUBROUTINE PARTL(NOVARS,WHICH,VAL,EPS,PARTYL,INCR,BASE)
C
      PARAMETER NOVALS=20,INFILE=5,OUTFIL=6
C
      DIMENSION VAL(NOVALS),VAL1(NOVALS)
C
      INTEGER NOVARS,WHICH,FIRST,I,J
C
      REAL VAL,VAL1,INCR,VALUE1,VALUE2,PARTYL,EPS,BASE
C
      FIRST=1
      INCR=1.
      DO 100 I=1,NOVARS
      VAL1(I)=VAL(I)
100   CONTINUE
      DO 200 J=1,50
      INCR=INCR*(.5)
      VAL1(WHICH)=VAL(WHICH)+INCR
      CALL PREPR(VAL1,VALUE1)
      IF (FIRST .NE. 1) GOTO 300
      PARTYL=(VALUE1-BASE)/INCR
      FIRST=0
      GOTO 200
300   IF (ABS(((VALUE1-BASE)/INCR)-PARTYL) .LT. EPS) RETURN
      PARTYL=(VALUE1-BASE)/INCR
200   CONTINUE
      RETURN
      END
```

5-8. PARTL SUBROUTINE FLOWCHART




2
Were fifty approximations made ?
No
3
Yes
Return

Section II. THE PREPR DRIVER SUBROUTINE

5-9. INTRODUCTION. This subroutine must be written by the user to provide an interface between the PROGTEST, PARTL, VARVARY1, GRID, and DIFFQUOT routines and the routine being tested. PREPR receives variable values from POINTCOMP or the other routines and returns the output value determined by calling the program being tested with the given variable values as input.

5-10. DISCUSSION. The input variable values are passed to PREPR in array V, in the same order as the values read in, i.e., the leftmost variable defined in the input is in V(1), etc.

5-11. PREPR LAYOUT

```
 SUBROUTINE PREPR(V,VALU)
 PARAMETER NOVALS = 20
 DIMENSION V(NOVALS)
 REAL V,VALU
.
.
Pass input values given in V to the program being tested.

Call program being tested as a subroutine using input values
passed in V.

VALU = value returned by the program.

END
```

APPENDIX A

CONTRIBUTORS

1. DOCUMMENTALIST

Dr. Steve Bravy, Methodology and Computer Support Directorate

2. TECHNICAL SUPPORT PERSONNEL

Mr. Joseph M. Tessmer

3. OTHER SUPPORT PERSONNEL

Ms Carrie Allen, Word Processing Center

Ms Julie Fuller, Word Processing Center

Ms Bobbie Guenthner, Word Processing Center

Ms Joyce Garris, Word Processing Center

SSG Richard Loller, Graphic Arts Branch

Ms Linda Prieto, Word Processing Center

APPENDIX B

DISTRIBUTION

Addressee	# of Copies
Defense Technical Information Center Building 5 Cameron Station Alexandria, VA 22314	2
The Army Central Library (ASDIRS) Room 1A600 Pentagon Washington, DC 20310	1
Federal Software Exchange Center 5285 Port Royal Rd. Springfield, VA 22161	1
National Technical Information Service Springfield, VA 22161	1
Command and Control Technical Center (CCTC) Room BE685 ATTN: Technical Library Pentagon Washington, DC 20301	1
Dept of Defense Computer Institute Technical Library Washington Navy Yard Building 175, Room 37 Washington, DC 20374	1
Assistant for Automation OJCS Room 2A932 ATTN: CPT Goddard (Navy) Pentagon Washington, DC 20301	1
Defense Sciences Office, Cybernetics Technology Division (DARPA) DSO/CTD 1400 Wilson Blvd Arlington, VA 22209	1

Defense Communication Engineering Center R801 (Lt. Col. Tufts) 1860 Wichle Ave. Reston, VA 22090	1
Defense Communication Agency Command Control Technical Center/C440 11440 Isaac Newton Square, North Reston, VA 22090	1
Director, Defense Nuclear Agency ATTN: NASD Lt. Cdr Gladwin Washington, DC 20305	1
Defense Advanced Research Project Agency (DARPA) Cybernetics Technology Division ATTN: Dr. Fields 1400 Wilson Blvd Arlington, VA 22209	1
Commander US Army and Electronic Research and Development Command Fort Monmouth, NJ 07703	1
Director US Army TRADOC Systems Analysis Activity ATTN: ATAA-SL ATAA-T White Sands Missile Range, NM 88002	2
Director US Army Tank-Automotive Research and Development Command Warren, MI 48090	1
Commander US Army Mobility Equipment Research and Development Command Fort Belvoir, VA 22060	1
Commander US Army Computer Systems Command Fort Belvoir, VA 22060	1

Commander US Army Natick Research and Development Cmd Natick, MA 01760	1
Chief, Defense Logistics Studies Information Exchange US Army Logistics Management Center Fort Lee, VA 23801	2
Commander US Army Logistics Center Fort Lee, VA 23801	1
Commander Army TRADOC Systems Analysis Activity White Sands Missile Range White Sands, NM 88002	1
Office of the Deputy Chief of Staff for Research, Development and Acquisition Headquarters, Department of the Army Pentagon Washington, DC 20310	1
Director of Army Automation Office Chief of Staff, Army Pentagon Washington, DC 20310	1
Commander Rock Island Arsenal ATTN: Technical Library Rock Island, IL 61201	1
Commander US Army Armament Research and Development Cmd. Dover, NJ 07801	1
Commander Harry Diamond Laboratories 2800 Power Mill Rd Adelphi, MD 20783	1

Commander
US Army Test & Evaluation Command 1
Dugway Proving Ground, MD 84022

Commander
US Army Aviation Research and Development Cmd 1
P.O. Box 209
St. Louis, MO 63166

Commander
Army Training and Doctrine Command 1
Fort Monroe, VA 23651

Commander
Army Material Systems Analysis Agency 1
Aberdeen Proving Ground
Aberdeen, MD 21005

Commander
Computer Systems Command 1
Fort Belvoir, VA 22060

Office of the Deputy Under Secretary of the Army
for Operations Research 1
ATTN: Mr. Dick Lester
Pentagon Room 2E621
Washington, DC 20310

Communications Research and Development Command
Center for Tactical Computer Systems 1
Division for Software Engineering
ATTN: Mr. Joe Kernan
Fort Monmouth, NJ 07703

Ballistic Research Laboratory
Building 393 1
ATTN: Mr. Jerry Thomas
Aberdeen Proving Ground
Aberdeen, MD 21005

Ballistic Research Laboratory Technical Library
Building 305 1
Aberdeen Proving Ground
Aberdeen, MD 21005

US Army Computer Systems Command
Command Technical Library H9 1
Fort Belvoir, VA 22060

US Army Management Systems Support Agency
DA-ACAM-DPD-E 1
ATTN: Jim Kilgore
Room BD1022
Pentagon Washington, DC 20310

Army Insitiute for Research in Management Information
and Computer Science (AIRMICS) 1
3131 Calculator Building
ATTN: John Mitchell
Georgia Tech
Atlanta, GA 30332

US Army Computer Systems Selection and
Acquisition Agency 1
Room 284
Hoffman I
ATTN: ACSA-TD-R
2461 Eisenhower Ave.
Alexandria, VA 22331

Chief of Naval Operations (OP-966D)
Department of the Navy 1
Washington, DC 20350

Director, Technology and Training Depot
Naval Data Automation Command 1
Washington Navy Yard
Washington, DC 20374

Headquarters, Dept of the Navy
Naval Material Command 1
ATTN: Mr. Owen McOmber MAT-084
Washington, DC 20360

Naval Data Automation Command (Code 40)
Washington Navy Yard 1
ATTN: Cpt Woodward
Washington, DC 20374

Office of Naval Research
Western Regional Office 1
ATTN: R. Lau
1030 East Green St.
Pasadena, CA 91106

Dr. Robert E. Conley
Office of the Chief of Naval Operations 1
OP-094H
Dept of the Navy
Room 4C679
Pentagon
Washington, DC 20350

Headquarters, USAF (AF/SAT)
Pentagon 1
Washington, DC 20330

Headquarters, Air Force
ATTN: Mr. Oscar Goldfarb 1
SAF-ALG
Washington, DC 20330

Air Force Geophysics Laboratory
ATTN: COL James Baker 1
Hanscom Air Force Base, MA 01731

Air Force Weapons Laboratory, Computer Branch (AFWL/AD)
ATTN: David McIntyre 1
Kirtland Air Force Base
Albequerque, NM 87117

Air Force Test & Evaluation Center, Software Branch
ATTN: LtCol Arner 1
Kirtland Air Force Base
Albequerque, NM 87117

Rome Air Development Center
ATTN: Alan Barnum 1
IS
Griffiss Air Force Base, NY 13441

Mr. Glen Ingram Scientific Computing Division Room A151 Technology Building National Bureau of Standards Washington, DC 20234	1
US Army Concepts Analysis Agency	
Chief, AD, ATTN: Library Branch	3
Director, FA	1
Director, JF	1
Director, RQ	1
Director, SM	1
Director, MC	25